INCREMENTAL APPROACH TO PARTICLE SWARM ASSISTED FUNCTION OPTIMIZATION

MO WENTING

NATIONAL UNIVERSITY OF SINGAPORE

2007

INCREMENTAL APPROACH TO PARTICLE SWARM ASSISTED FUNCTION OPTIMIZATION

Submitted by

MO WENTING

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF

ELECTRICAL & COMPUTER ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2007

Acknowledgments

There are many people whom I wish to thank for the help and support they have given me throughout the course of my Ph.D. program. My foremost thank goes to my supervisors. I thank Dr. Sheng-Uei Guan for his insights and suggestions that helped to shape my research skills. I thank Dr. Sadasivan Puthusserypady for his patience and encouragement that carried me on through all the difficult times. Their valuable feedback contributed greatly to my research work, definitely including this thesis.

Furthermore, I am thankful to Dr. Fangming Zhu and Ms. Qian Chen, for their kindness and help at the beginning of my Ph.d. course. They set good examples for me as a research scholar, and their visionary thoughts inspired me on my research topics.

Last but not least, I would like to thank my parents for always being there when I needed them most, and for supporting me through all these years. I would especially like to thank my boyfriend Weijia for his love and support. This dissertation is dedicated to them.

Publications Originated from my PhD Work

Journals:

1. Sheng-Uei Guan, Qian Chen and Wenting Mo, "Evolving Dynamic Multi-Objective Optimization Problems with Objective Replacement," *Artificial Intelligence Review*, vol. 23, pp. 267-293, 2005.

2. Sheng-Uei Guan, Qian Chen and Wenting Mo, "An Evolutionary Strategy for Decremental Multi-Objective Optimization Problems," *International Journal of Intelligent Systems*, vol. 22, no. 8, pp. 847-866, 2007.

3. Sheng-Uei Guan and Wenting Mo, "Incremental Evolution Strategy for function optimization," *International Journal of Hybrid Intelligent Systems*, vol. 3, no. 4, pp. 187-203, 2006.

4. Wenting Mo, Sheng-Uei Guan and Sadasivan Puthusserypady, "An Incremental Optimization Model for Function Optimization," *Machine Learning*, communicated (under third review).

5. Wenting Mo, Sheng-Uei Guan and Sadasivan Puthusserypady, "Ordered Incremental Multi-Objective Problem Solving," *Artificial Intelligence Review*, communicated (under review).

Book Chapter:
Wenting Mo and Sheng-Uei Guan, "*A Novel Hybrid Algorithm for Function optimization*," Hybrid Evolutionary Systems, Springer Berlin, vol. 75, pp. 101-125, 2007.

Conference:

Wenting Mo, Sheng-Uei Guan and Sadasivan Puthusserypady, "Particle Swarm Assisted Incremental Evolution Strategy for Function Optimization," In *Proceedings of the IEEE International Conference on Cybernetics and Intelligent Systems*, vol. 3, pp. 187-203, Bangkok, Thailand, June, 2006.

List of Abbreviations

ACO	Ant Colony Optimization
ASPSO	Asynchronous version of PSO
BBS	Bulletin Board System
BFGS	Broyden-Fletcher-Goldfarb-Shanno
CB	Combined Balance
CCGA	Cooperative Coevolutionary GA
CCL	Cumulative Conflict Level
CF	Correlated Function
CIA	Computationally Intelligent Algorithms
CL	Conflict Level
CLOOA	Conflict level based Objective Ordering Approach
CPSO	Cooperative PSO
CSPSO	PSO with Charged Swarm
DFP	Davidon-Fletcher-Powell
DOOA	Difficulty based Objective Ordering Approach
EA	Evolutionary Algorithm
EMOO	Evolutionary Multi-Objective Optimization
EP	Evolutionary Programming
ES	Evolution Strategy
FR	Fletcher-Reeves
GA	Genetic Algorithm

GP	Genetic Programming
HPSO_BS	Hybrid PSO with Breeding and Subpopulations
ICP	Ideal Cutting Planes
IGO	Incremental Global Optimization
IMOGA	Incremental MOGA
IMOO	Incremental MOO
IMOPSO	Incremental MOPSO
IPSO	Incremental PSO
MOEA	Multi-Objective Evolutionary Algorithm
MOGA	Multi-Objective GA
MOO	Multi-Objective Optimization
MOP	Multi-Objective Problems
MOPSO	Multi-Objective PSO
MVO	Multi-Variable Optimization
NFL	No Free Lunch
NSGA-II	GA with Non-Dominated Sorting
OCP	Optimal Cutting Plane
PAES	Pareto Archived ES
PIPSO	Parallel IPSO
PR	Polar-Ribiere
PSO	Particle Swarm Optimization
QoS	Quality of Services
RA	Relative Accuracy
SB	Sequential Balance
SI	Swarm Intelligence
SOO	Single Objective Optimization
SOP	Single Objective Problems
SPEA	Strength Pareto EA
SVO	Single Variable Optimization
UF	Uncorrelated Functions

Contents

Summary

The need for optimization exists in almost every aspect of our daily life. Many economic, scientific and engineering problems require adjusting parameters to achieve a more desirable outcome in one or more objectives. A variety of techniques have been developed for solving either the single objective problems (SOPs) or the multi-objective problems (MOPs). They are categorized broadly into deterministic algorithms, stochastic algorithms and intelligent algorithms. In the last two decades, the intelligent algorithms have received vast interest for their "intelligence" exhibited in solving problems with various difficulties, such as large number of local optima, and landscapes with ridges and deceptive flatness. However, as the dimensionality of those problems increases, their complexity may increase exponentially. In these cases, the intelligent algorithms may tend to be trapped in local optima or arbitrary points. In order to improve the ability of these intelligent algorithms in handling such high dimensional problems, this thesis studies incremental optimization techniques. The investigation focuses mainly on two aspects:

1. The incremental optimization in the input space, which incrementally optimizes the variable set for global optimization problems. The investigation resulted in the following contributions:

    - The feasibility of the incremental global optimization (IGO) is proved mathematically. Based on this, an incremental model in the input space

is proposed, which allows the intelligent algorithms to optimize from low dimensional spaces and then progress to higher dimensional spaces incrementally. The IGO could benefit the standard intelligent algorithms in terms of increasing their global convergence probability.

- Particle Swarm Optimization (PSO) is used as a vehicle to demonstrate the advantages of the incremental model, resulting in a novel PSO-based IGO algorithm, the Incremental PSO (IPSO). Experiments on IPSO have shown that PSO could profit from using the proposed model.
- A parallel version of IPSO is designed in order to increase its efficiency of information sharing. In the parallel IPSO (PIPSO), the information obtained from different search spaces with reduced dimensionality is collected and broadcasted through a Bulletin Board System (BBS). With this information sharing mechanism, the PIPSO is able to outperform the IPSO in the experiments on several benchmark problems.

2. The incremental optimization in the output space, namely the incremental multi-objective optimization (IMOO), which incrementally optimizes the objective set for MOPs. The main contributions are listed below:

   - For IMOO, the relationship between the Pareto fronts before and after objective increment is analyzed. One theorem and two corollaries are proved to state the rationale behind the IMOO. Based on this rationale, an incremental model in the output space is built for multi-objective intelligent algorithms to obtain more desirable Pareto-optimal solutions.
   - As a relatively new "intelligent multi-objective optimization algorithm", multi-objective particle swarm optimization (MOPSO) is chosen to be the vehicle to show the efficacy of the incremental model built in the output space. By applying the model to MOPSO, a novel PSO-based IMOO, Incremental Multi-Objective Particle Swarm Optimization (IMOPSO)

is implemented. Experiments on IMOPSO have shown that MOPSO could benefit from using the incremental model in the sense of obtaining "better" Pareto fronts.

- An important issue of IMOO, the impact of objective ordering, is explored. An objective ordering approach is proposed, which aims at obtaining the optimal objective order so that an IMOO algorithm can achieve its potential best performance.

List of Figures

List of Tables

Chapter 1

Introduction

When you read a daily newspaper, has it ever crossed your mind that the circulation of the newspaper was optimized to maximize the business profit of the newspaper? When you call your friends (by telephone), can you imagine how many variables in the network were configured to maximize the Quality of Services (QoS) as well as the profit of the telecom service provider? The need for optimization exists in almost every aspect of our daily life. Many economic, scientific and engineering problems require adjusting the parameters to achieve a more desirable outcome in one or more objectives. These problems can be categorized into: (i) Combinatorial problems that have a linear or nonlinear function defined over a finite but very large set of solutions, (ii) General unconstrained problems that have a nonlinear function over reals that are unconstrained (or have simple bound constraints) and (iii) General constrained problems that have a nonlinear function over reals that are constrained by some other functions. The study presented in this thesis focuses on effectively solving the general unconstrained optimization problems, which could be a base for further studies on the other categories of optimization problems. So, the "optimization problems" mentioned in this thesis refer to the general unconstrained optimization problems. The motivation for the present study and the main

objectives are detailed in the following subsections.

1.1 Motivation of Research

In the general unconstrained optimization problems, the problems with only one objective are single objective problems (SOPs), while the others with several objectives (that may be conflicting) are multi-objective problems (MOPs). *Global optimization* solves SOPs by adjusting the parameters to obtain the global best output. Multi-objective optimization, on the other hand, handles MOPs by looking for tradeoff solutions. Regarding the tradeoff solutions, none of the objectives can be improved without compromising the other objectives. A variety of techniques have been developed for solving both SOPs and MOPs. For global optimization, the approaches can be categorized roughly into three groups, including deterministic algorithms, Monte-Carlo-based stochastic algorithms and computationally intelligent algorithms (CIAs). The deterministic algorithms solve global optimization problems precisely, but they may rely on the availability of an analytic formulation of the objective function (e.g. interval methods [1]). The Monte-Carlo-based stochastic algorithms (e.g. simulated annealing [2, 3]) usually start from a random initial solution and improve the solution by stochastic approximation. The CIAs (e.g. evolutionary algorithms and swarm intelligence) perform the searching in an intelligent way and are mostly population-based.

An evolutionary algorithm (EA) uses some mechanisms inspired by biological evolution: *reproduction*, *mutation*, *recombination*, *natural selection* and *survival of the fittest* [4–6]. The well-known EAs include genetic algorithm (GA), evolution strategy (ES), genetic programming (GP) and evolutionary programming (EP). Swarm Intelligence (SI) is the property of a system whereby the collective behaviors of (unsophisticated) agents interacting locally with their environment cause the

coherent functionally global patterns to emerge [7, 8]. This kind of systems can be found in nature, such as ant colonies, bird flocking, fish schooling etc. The well-known SI technologies include ant colony optimization (ACO) and particle swarm optimization (PSO). Since these heuristic algorithms do not require any prior knowledge about the problem or make any assumption about the underlying fitness landscape, they have been increasingly used in the past several decades.

For multi-objective optimization, the focus is not to develop new types of algorithms, but to extend or revise the algorithms used for single objective optimization to solve MOPs. Many researchers convert the multiple objectives into a single objective by assigning a weight to each individual objective and then using deterministic algorithms to solve this converted SOP. With the development of heuristic algorithms, more and more researchers realize the merits of employing them to solve MOPs. Many multi-objective optimization algorithms have been developed based on the biologically inspired heuristic algorithms, including Multi-objective Evolutionary Algorithms (MOEAs) [9] and Multi-objective Particle Swarm Optimization (MOPSO) [10]. The commonly prominent characteristic of these multi-objective optimization algorithms is that they can find a set of tradeoff solutions, named Pareto-optimal set, in a single run.

For optimization problems, the variables that need to be adjusted form the *input space* while the objective(s) that need to be optimized form the *output space*. Generally, the heuristic optimization algorithms treat the input space and the output space as a whole. In this case, they may not be able to obtain satisfactory performance when the dimensionality of the input/output space is high, or the variables are highly coupled, or the objectives are seriously conflicted. Hence, this thesis investigates an incremental approach for solving unconstrained optimization problems, which handles the variables or the objectives incrementally. The incremental approach is based on the hypothesis that the solutions found in subproblems, formed by projecting the original problem into subspaces with reduced

dimensionality, may keep their superiority to some extent. According to this hypothesis, incremental models are designed to conduct searching from subspaces with lower dimensionality to those with higher dimensionality, with the found solutions being inherited. They are built in the input and the output spaces, respectively, to be applied to CIAs. The resulting incremental CIAs used the information collected in lower-dimensional subspaces to guide the search in higher-dimensional subspaces, with the dimensionality increasing one by one. Since solving problems with lower dimensionality is easier compared to solving those with higher dimensionality and the inheritance mechanism makes the latter easier, it is expected that the proposed incremental CIAs would improve the performance of the original CIAs, especially when dealing with complicated optimization problems. In particular, PSO, a relatively new CIA, is employed for its flaw when scaled (discussed in detail in Chapter 3) as a vehicle to study the characteristics of the incremental models.

1.2 Objectives and Scope of Research

The overall goal of the present study was to propose incremental models in both the input and output spaces for CIAs to expand their capacity to solve complicated optimization problems with highly coupled variables, or a large number of variables or conflicting objectives. The main objectives and milestones are listed below:

1. *Incremental model in the input space*

   This study investigated an innovative incremental technique for global optimization. This technique was aimed to improve the performance of the conventional CIAs in terms of the rate of convergence and the quality of solution. The milestones include:

   (a) Theoretical analysis was provided for performing incremental optimization. An incremental model was built based on the conclusion.

(b) Since the scalability problem of PSO (one of the CIAs) is well-accepted, it was chosen as a vehicle to study the effect of equipping the conventional CIAs with the incremental technique. The resulting incremental PSO (IPSO) optimizes an objective function from single variable, followed by inheritance, integration and further optimization with more variables concerned.

(c) A parallel model was developed based on the IPSO to further improve its ability of dealing with global optimization problems with highly coupled variables, and to exhibit the possibility of parallel implementation as well.

2. *Incremental model in the output space*

This study further investigated the incremental technique in the output space of optimization problems, resulting in incremental multi-objective optimization. The multi-objective optimization involves more than one objective functions, which can not achieve their optimal values with the same decision vector. The milestones include:

(a) Theoretical foundation for performing incremental multi-objective optimization was given by analyzing the relationship between the Pareto fronts obtained before and after objective increment. Based on the analysis, an incremental model was built.

(b) For the continuity and systematization of the study, the multi-objective version of PSO, MOPSO in short, was chosen to be the vehicle to examine the usefulness of the incremental technique.The resulting incremental MOPSO (IMOPSO) will optimize the objective set from single objective, followed by inheritance, integration and further optimization with more objectives involved.

(c) The impact of objective ordering on the performance of incremental

multi-objective optimization was investigated in order to obtain an approach, which provides an optimal order to arrange the objectives with affordable computation cost.

Generally speaking, the presented incremental techniques for global optimization and multi-objective optimization may be useful to overcome some drawbacks of CIAs, including premature convergence and poor scalability. Besides, the theoretical analysis of incremental models would shed light on understanding why CIAs benefit from the incremental techniques when solving optimization problems. Moreover, this study could stimulate further an interest in developing incremental techniques in other engineering fields as well.

In this thesis, the incremental techniques are proposed and developed for enhancing the performance of CIAs, which differ from conventional deterministic optimization techniques in various aspects (as stated in Chapter 2). Thus, the incremental technique is discussed in the context of CIAs. The comparison of the proposed incremental algorithms with the deterministic optimization techniques is beyond the scope of this study. Also, the investigation of the incremental techniques is restricted to the general unconstrained problems, which would form a base for further studies on the other categories of optimization problems.

1.3 Methodology

This thesis mainly focuses on incremental global optimization and incremental multi-objective optimization. For each topic, there is a fixed investigation flow described as follows:

1. Theoretical analysis is provided to state the rationale and support the feasibility of the corresponding incremental model.

2. Details about how to implement a PSO-based incremental model are described. In this thesis, PSO is employed as the vehicle to present the profits of using the incremental models. However, it does not mean that the incremental models are particularly designed for PSO. On the contrary, the incremental models are suitable for almost all the CIAs, such as the GA and ES. This is because the components of an incremental model can be implemented by any CIA, and the integration between two components is solution-based, which is independent of algorithm. In chapter 3, we provide a hybrid implementation of the incremental model in the input space to support this statement.

3. Experiments are conducted on various synthetic benchmark problems with well-known features. Performance comparisons are made with standard and improved CIAs to show that incremental models could make the standard CIAs obtain better performance without extra efforts of revising the algorithms. However, the issue of whether the CIAs are "better" than other methods or whether one CIA is "better" than another from a computational perspective is not the focus of this thesis.

4. Some specialized issues (such as the information sharing mechanism and the objective ordering issue) are considered and discussed so that we can gain more insight into the incremental models.

1.4 Contributions of this Study

The following are the main contributions of this thesis:

1. We have proved the feasibility of incremental global optimization, followed by building an incremental model for CIAs in the input space. This model

allows CIAs to optimize from low dimensional spaces and then move to higher dimensional spaces incrementally. The incremental optimization in the input space could benefit the CIAs in terms of increasing their global convergence probability.

2. We have designed and implemented a novel PSO-based incremental global optimization, IPSO, based on the incremental model built in the input space. Experiments on IPSO have shown that PSO could profit by using the input space incremental model.

3. A hybrid implementation of the incremental model by using both PSO and (1+1)-ES has been studied. This hybrid incremental algorithm has been shown experimentally more efficient than the pure PSO-based incremental algorithm, the IPSO.

4. The parallelizability of the IPSO has been investigated by designing a parallel version of it. With this parallel IPSO (PIPSO), the information gained from searching in the spaces with reduced dimensionality could be shared with a higher efficiency.

5. We have analyzed the relationship between the Pareto fronts before and after objective increment and concluded the rationale behind the incremental optimization in output space, i.e. incremental multi-objective optimization. Based on this rationale, an incremental model in the output space has been built for multi-objective CIAs to obtain more satisfying Pareto-optimal solutions.

6. A novel PSO-based incremental multi-objective optimization, IMOPSO, has been designed and implemented based on the incremental model built in the output space. Experiments on IMOPSO have shown that MOPSO could benefit by using the output space incremental model.

7. We have also investigated the impact of objective ordering on the incremental multi-objective optimization. Considering the impact, we have proposed an objective ordering approach. This approach aims at finding the optimal objective order so that the incremental multi-objective optimization achieve its potential best performance.

1.5 Outline of the Thesis

Chapter 2 provides sufficient background knowledge of the optimization, as well as the problem definitions. It also provides a review of the research works that have close relationship with our study. Chapter 3 focuses on the incremental optimization in the input space. It starts with the graphical analysis and proof of the feasibility and rationale of incremental global optimization, followed by proposing an incremental model designed for global optimization. This is again followed by employing PSO as a vehicle to present the profits by using the incremental model. In addition, (1+1)-ES is used together with PSO to realize a hybrid implementation of the proposed incremental model. Chapter 4 explores a parallel version of the incremental global optimization as a supplementary model. As the vehicle used to show the benefits of incremental optimization both in the input and output spaces, PSO plays an important role in the thesis. Chapter 5 focuses on incremental optimization in the output space. It starts with the theoretical analysis of the effect of objective increment on the Pareto front, followed by proposing an incremental model designed for multi-objective optimization. This is followed by employing PSO as a vehicle to present the profits by using the incremental model. Chapter 6 discusses the objective ordering issue of the incremental multi-objective optimization model. Factors influencing the performance of the model which are associated with the objective order are analyzed. Based on the analysis and the experimental results, an objective ordering approached is proposed. Chapter 7 summarizes the

findings of this thesis, along with topics for future research are also given.

In view of the objectives mentioned in this chapter, it may be noted that the CIAs form the basis of this study. The widely used CIAs and some of the well known modifications to them are reviewed in the following chapter.

Chapter 2

Background and Related Work

This chapter introduces some of the basic definitions used in optimization, and clearly highlights the problems investigated in later chapters of this thesis. A brief introduction to Evolutionary Algorithms (EAs) is provided, and the major issues related to GAs and ESs, which are the most important algorithms in EAs history, are addressed. Subsequently, the PSO is introduced and its flaws is discussed. Besides, this chapter reviews previous work related to this thesis.

2.1 Optimization Problems

The optimization problems with only one objective function are called SOPs. In contrast, some optimization problems may involve more than one objective functions, and they are called MOPs. Many algorithms were developed to handle SOPs and MOPs, which are reviewed in this section. What should be noted is that the term optimization refers to both minimization and maximization tasks. Actually, the terms minimization, maximization and optimization shall be interchangeable, as the task of maximizing the objective function f is equivalent to minimizing $-f$.

Without the loss of generality, only minimization tasks were used in the study presented in this thesis.

2.1.1 Single Objective Optimization

Single objective optimization involves finding the best solution to a given problem. This task is of great importance to many professions [11]. For example, the communication engineers use optimization techniques to find optimal parameters in antennas design to obtain satisfactory transmission/receiving efficiency. Biological scientists require optimization algorithms when performing Deoxyribonucleic acid (DNA) microarray data analysis to discover useful guidelines to distinguish genes associated with various cancers. Logistics researchers have to consider the optimal allocation of resources in logistic configuration. Mathematically, the single objective optimization is to find the minimum or maximum of a function that is called the *objective function* of a SOP. The objective functions can be categorized by various criteria such as linear-or-not and constrained-or-not. The optimization problems with linear objective functions are called *linear optimization problems*, which can be solved efficiently by a technique known as the linear programming [12]. The others are known as *non-linear optimization problems*, which are generally very difficult to solve. By analogy, the optimization problems with objective functions subject to certain constraints are called *constrained optimization problems*, while the others are called *unconstrained optimization problems*. In this thesis, we are concerned with unconstrained non-linear optimization problems with continuous variables. Their mathematical definitions are given below.

Basic Definitions

The general form of the SOP dealt with in this thesis is defined as:

- *Objective function*: $f(x_1, \ldots, x_d)$, where d may be any integer greater than zero.
- *Search (input) space*: $S \subseteq \mathbb{R}^d$.
- *Task*:

$$\min\{f(\mathbf{x}) \mid \mathbf{x} \in S\}, \tag{2.1}$$

where x_i $(i = 1, \ldots, d)$ is called the *decision variable* of f, and $\mathbf{x} = [x_1, x_2, \ldots, x_d]$ is named the *decision vector*.

Following the definitions given in [13], the point $\mathbf{x}^*_\varepsilon$ in the region S is said to be a *local minimizer* of the function $f(\mathbf{x})$, subject to $\mathbf{x} \in S$, if there exists a small positive number ε such that

$$f(\mathbf{x}^*_\varepsilon) \leq f(\mathbf{x}), \tag{2.2}$$

for all $\mathbf{x} \in S$ which satisfy $\|\mathbf{x}^*_\varepsilon - \mathbf{x}\| \leq \varepsilon$. The value of $f(\mathbf{x}^*_\varepsilon)$ is then the corresponding *local minimum*. The norm, or the distance measure, is the Euclidean norm, and is defined in Equation (2.3) below:

$$\|\mathbf{x}\| = \sqrt{\sum_{i=1}^{d} x_i^2} \quad . \tag{2.3}$$

The point $\mathbf{x}^*$ is a *global minimizer* of the function $f(\mathbf{x})$, if

$$f(\mathbf{x}^*) \leq f(\mathbf{x}) \tag{2.4}$$

for all $\mathbf{x} \in S$. The value of $f(\mathbf{x}^*)$ is then the *global minimum* of the function $f(\mathbf{x})$ in the region S. Throughout the thesis, unless otherwise stated it is always understood that the minimum required in a SOP is a global one. Often we assume S is a closed convex set and $f(\mathbf{x})$ is continuous on S, but occasionally the case where $f(\mathbf{x})$ is discontinuous at certain points of S also is discussed, especially when this turns out to be important for evaluating the algorithms.

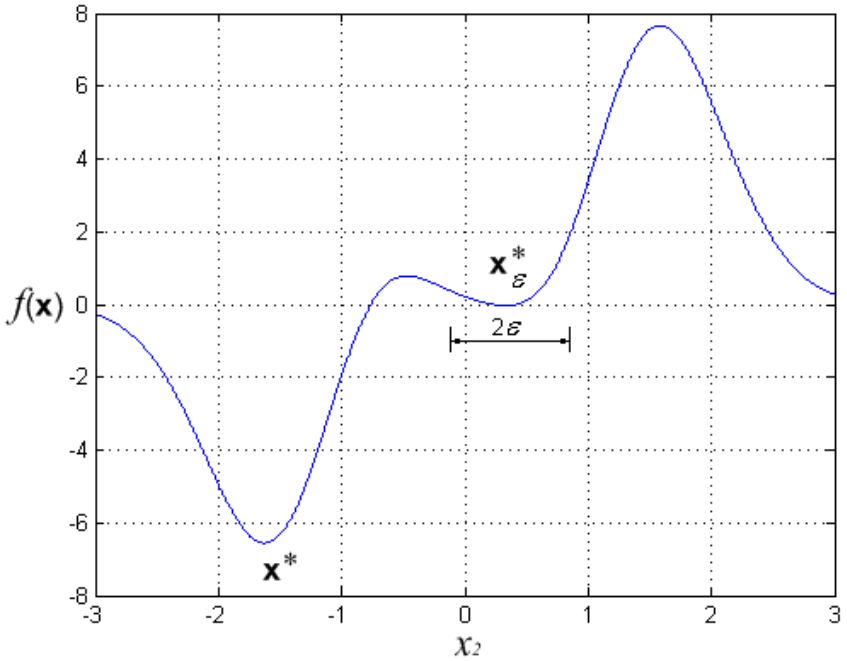


Figure 2.1: Landscape of the function in Equation (2.5), indicating the global *vs.* local minimizer

Local *vs.* Global Search

Figure 2.1 illustrate the difference between the local minimizer $\mathbf{x}_\varepsilon^*$ and the global minimizer $\mathbf{x}^*$ for the landscape function given in Equation (2.5).

$$\begin{aligned} f(x_1, x_2) =& 3(1-x_1)^2 \exp(-x_1^2 - (x_2+1)^2) \\ & - 10(\frac{x_1}{5} - x_1^3 - x_2^5) \exp(-x_1^2 - x_2^2) - \frac{1}{3}\exp(-(x_1+1)^2 - x_2^2), \end{aligned} \tag{2.5}$$

with $x_1 = 0.23$ and $x_2 \in [-3, 3]$.

Local search is concerned with converging to a local minimizer when solving a SOP. Normally, local search algorithms start from a candidate solution $\mathbf{x}_0 \in S$ and then iteratively try to improve upon it by moving to neighbor solutions with decreasing objective value, generally known as the hill-climbing algorithms. There has been an immense amount of work in the past on local search algorithms for unconstrained optimization [12–14], including deterministic algorithms and stochastic algorithms.

The deterministic local search algorithms are usually concerned with the derivatives of objective function, e.g. simple Newton-Raphson algorithms and its many variants, including the scaled conjugate gradient algorithm [15] and the quasi-Newton [12, 15] family of algorithms. Some of the well known deterministic local search algorithms include Fletcher-Reeves (FR), Polar-Ribiere (PR), Davidon-Fletcher-Powell (DFP) and Broyden-Fletcher-Goldfarb-Shanno (BFGS) [12, 14]. The drawback of these algorithms, which rely on such concepts as derivatives, gradients, subdifferentials and the like, is that their performance depends largely on the position of their starting (initial) points. They are likely to converge to the local minimizer close to the starting points. Strictly speaking, they may even fail to locate a local minimizer because of saddle points [16], shown in Figure 2.2.

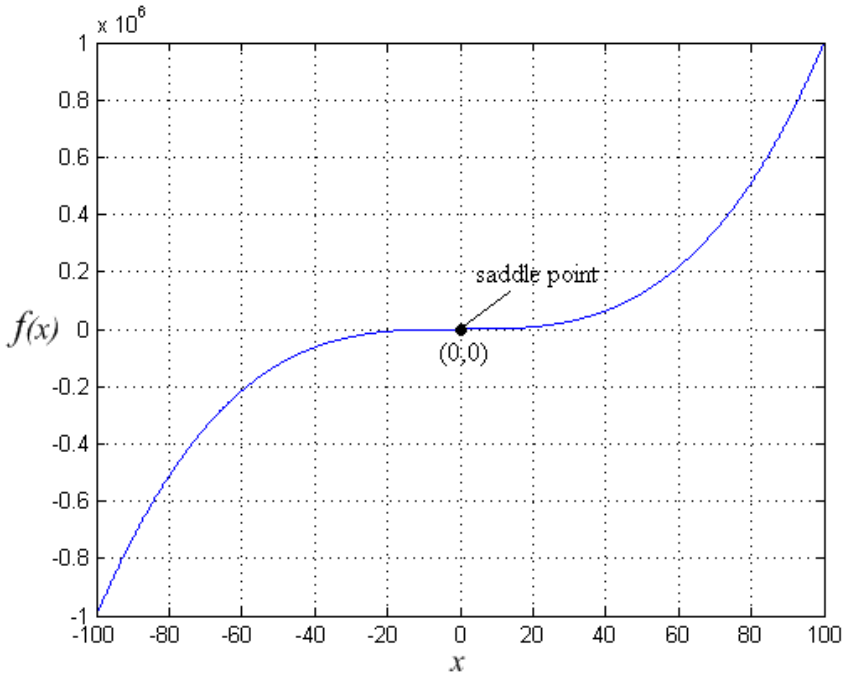


Figure 2.2: Plot of $f_(x) = x^3$ with a saddle point at (0,0)

The stochastic local search algorithms try to overcome the flaws of the deterministic algorithms mentioned above, including stochastic hill climbing, random walks and simulated annealing (SA) [3, 17, 18]. These algorithms dramatically reduce the dependence on the position of the initial solution. However, they may still have premature convergence problems when dealing with SOPs of certain dif-

ficulties, such as ridges and plateaus [19]. A ridge is a curve in the search place that leads to a minimum, but the orientation of the ridge compared to the available moves that are used to climb is such that each move will lead to a point with larger objective value. In other words, each point on a ridge looks to the algorithm like a local minimum, even though the point is part of a curve leading to a better optimum. Another problem with hill climbing is that of a plateau, which occurs when we get to a "flat" part of the search space, i.e. we have a path where the heuristics are all very close together. This kind of flatness can cause the algorithm to cease progress and wander aimlessly.

In contrast to local search, global search requires to find the global minimizer. Theoretically, the core of global search is to address the fundamental question that how to check whether a solution is global optimal, and if it is not, find a better feasible solution. However, the checking is mathematically impossible when the second derivatives of objective function are not available. That is to say, there hardly exist strict global search algorithms, as discussed in the two collections of papers on the topic of true global optimization algorithms edited by Dixon and Szego [20, 21]. Since it is the possibility to become trapped at a stationary point which causes the failure of local search algorithms and motivates the need to develop global search algorithms, the essential spirit of global search is to reduce the probability of premature convergence. Evolutionary algorithms discussed in Section 2.2 are generated under this guideline, which are characterized with an intensive use of randomness and genetics-inspired operations to evolve a set of candidate solutions.

2.1.2 Multi-Objective Optimization

Real-world optimization problems often require the minimization/maximization of more than one objective, which, in general, conflict with each other. The root

of these MOPs (alternatively vector optimization problems) can be seen in the theory of games as a branch of economics [22], where the links to multi-objective optimization are the study of minimax theorems, games with vector payoffs and equilibrium points. The mathematical description of an MOP is given below.

Basic Definitions

From a mathematical point of view, an MOP with d decision variables and n objectives aims to find which minimizes the values of the objective functions within the feasible input space S, which is stated in its general form as follows:

- *Objective set*: $F = \{f_1, \ldots, f_n\}$, where n may be any integer greater than one.
- *Input space*: $S \subseteq \mathbb{R}^d$.
- *Task*:

$$\min\{f_i(\mathbf{x}) \mid \mathbf{x} \in S\}, \quad i = 1, \ldots, n. \tag{2.6}$$

where x is the same with the decision vector defined previously.

One of the striking differences between SOP and MOP is that in MOP the objective functions constitute a multi-dimensional space, in addition to the usual decision variable space (*input space*) S. This additional space is called the *objective space* (*output space*), denoted as O. For each solution $\mathbf{x}$ in the input space, there exists a point $\mathbf{z}$ in the output space, denoted by $\mathbf{f}(\mathbf{x}) = \mathbf{z} = [z_1, z_2, \ldots, z_n]$, where $\mathbf{f} = [f_1, f_2, \ldots, f_n]$.

In MOPs, the presence of multiple objectives results in a set of optimal solutions (named the *Pareto-optimal set*), instead of only one global optimal solution as in

SOPs. The corresponding set of optimal points in the objective space is called *Pareto-optimal front*. For each solution in the Pareto-optimal set, no improvement can be achieved in any objective without degradation in at least one of the others. Without further information, one Pareto-optimal solution cannot be declared as better than another. Stated mathematically, let $\mathbf{x}, \mathbf{h} \in S$,
$\mathbf{x}$ is *dominated* by (or inferior to) $\mathbf{h}$ if

$$f_i(\mathbf{x}) \geq f_i(\mathbf{h}), \forall i \in [1, n] \quad \text{AND} \quad f_i(\mathbf{x}) > f_i(\mathbf{h}), \exists j \in [1, n], \tag{2.7}$$

and $\mathbf{h}$ is *Pareto-optimal* in S if there is no solution dominates it.

Non-Pareto *vs.* Pareto-based Multi-Objective Optimization

With regard to the fitness assignment, most multi-objective optimization methods fall into two categories, non-Pareto and Pareto-based [23, 24]. non-Pareto methods [25–27] directly use the objective values to decide an individual's survival. Schaffer's VEGA [26] is such an example. VEGA generates as many sub-populations as objectives, each trying to optimize one objective. Finally all the sub-populations are shuffled together to continue with genetic operations. In contrast, Pareto-based methods [24, 28–32] measures individuals' fitness according to their dominance property.

Usually a user needs only one solution, no matter whether the associated optimization problem is single-objective or multi-objective. Some traditional Non-Pareto methods, including weighted sum, ϵ-constraint and goal programming, search for only one optimal solution in each run, by using *a priori* articulation of the preferences to the objectives [33, 34]. However, deciding these preferences involves high-level information which is often hard to obtain beforehand. In contrast, if a set of many trade-off solutions are already worked out or available, one can evaluate

the pros and cons of each of these solutions based on his own considerations and compare them to make a final choice. That is why Pareto-based multi-objective optimization methods are used. The Pareto-based methods aim at finding the set of such trade-off solutions, namely Pareto-optimal solutions, by considering all objectives to be of the same importance.

Since evolutionary computation algorithms maintain a population of solutions, they can find a number of solutions distributed uniformly in the Pareto-optimal set in a single run with certain diversity maintenance mechanisms, which distinguishes them from the classical non-Pareto methods mentioned above. In other words, evolutionary computation is ideal for Pareto-based multi-objective optimization. Actually, a number of multi-objective evolutionary algorithms have been suggested [35, 36], which are discussed in Section 2.4.2.

2.2 Famous Evolutionary Algorithms for Optimization

As stated in Section 2.1.1, although the classical local search algorithms work well for some optimization problems, they may not scale well when facing tasks with difficulties such as numerous local optima, saddle points, ridges and plateaus, which usually turns worse with increasing dimensionality. By the intensive use of randomness and genetics-inspired operations to evolve a set of candidate solutions, EAs is a powerful search and optimization paradigm that is considered a global search method and a general problem solver [37]. They have become popular tools for search, optimization, machine learning and solving design problems. Historically, Genetic Algorithms (GAs), Evolution Strategies (ESs) and Evolutionary Programming (EP) are the prominent approaches in the EAs' family. All of them can be used for global optimization. GAs have long been viewed as multi-purpose tools

with applications in search, optimization, design and machine learning [38, 39]. Most of the work in ESs focused on real-valued optimization [28, 40, 41]. GP is famous for extending the genetic model of learning to the space of programs [42, 43]. Although this thesis mainly focuses on PSO-based function optimization, EAs, including GAs and ESs, are occasionally mentioned for comparison or discussion as supporting contents. Therefore, a brief introduction to GAs and ESs is given below. For a more comprehensive introduction to GAs and ESs, please see [44] and [40], respectively.

2.2.1 Genetic Algorithms

Although GAs are not the focus in this study, they have an extremely significant position in the area of intelligent algorithms, which will be inevitably mentioned in the later chapters of this thesis. So, their background knowledge is briefly introduced here. GAs were invented by John Holland [38]. As the name implies, they model the evolutionary process at the level of the genome. The main idea is that in order for a population of individuals to collectively adapt to some environment, it should behave like a natural system, in which survival, therefore reproduction, is promoted by the elimination of useless or harmful traits and by rewarding useful behavior.

To use GAs, a solution to an optimization problem must be represented as a genome (or chromosome), which is typically represented by a string with a two-letter alphabet consisting of ones and zeros. The GAs then create a population of solutions and applies genetic operators such as recombination and mutation to evolve the solutions in order to find the best one(s). As described in [38], the pseudo-code of the canonical GA is shown in Figure 2.3.

The recombination and mutation operations are described in detail as follows:

```
t = 0
Initialize P_t to random chromosomes from {1,0}^l
Evaluate fitness of chromosomes in P_t
WHILE (termination conditions is false)
{
        Select chromosomes for reproduction from P_t based on fitness
        Apply genetic operators to reproduction pool to produce offspring
        Evaluate fitness of offspring
        Replace members of P_t with offspring to produce P_{t+1}
        t = t+1
}
```

Legends: t is a discrete unit of time, P is a population and l is the chromosome length.

Figure 2.3: Pseudo-code of the canonical GA

1. *Recombination*

   The recombination operation recombines genetic material from two parents to produce offspring. The simplest and most commonly used recombination operator is the *single-point crossover* that picks a locus randomly in the bit string and exchanges all the bits after that locus between the two parents, as shown in Figure 2.4. The other two commonly used recombination operators are *Two-point crossover* and *uniform crossover*. The two-point crossover is slightly more conservative than the one-point crossover, which picks two random loci to demarcate the boundaries of the exchange, resulting on average in a smaller segment than that produced by the single-point crossover. Uniform crossover swaps each bit from one parent chromosome with the corresponding bit from another parent chromosome with a probability of 0.5. It is considered to be more disruptive than the two-point crossover operator [45].

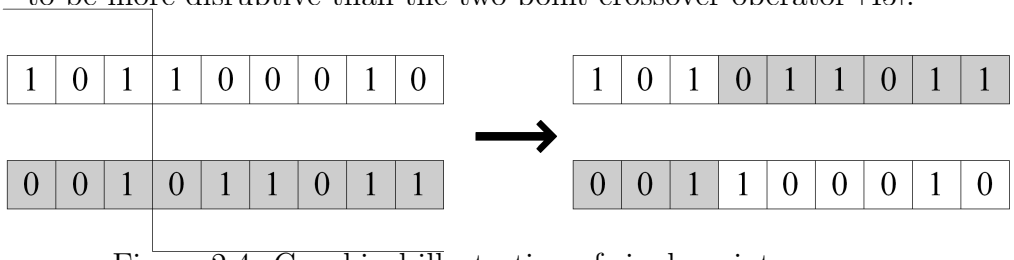


Figure 2.4: Graphical illustration of single-point crossover

2. *Mutation*

   The mutation operator as shown in Figure 2.5 randomly flips bits in a chro-

mosome to their opposite state. In GAs, mutation is normally applied at a low rate and a typical mutation rate is $1/l$, where l is the number of bits in the chromosome [44]. This results in an average of one bit mutated in each chromosome.

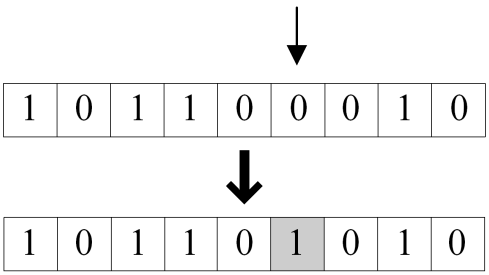


Figure 2.5: Graphical illustration of the mutation

One important distinction between mutation and recombination is that mutation has the ability to introduce new alleles into the genetic pool, while recombination does not have this effect, which only recombines existing genetic materials. In other words, using mutation favors exploration, while applying recombination promotes exploitation.

What should be noted is that although the notion of bit string encoding, namely genotypic representation, used to be one of the defining difference between GAs and other types of EAs, this distinction has blurred somewhat with the advent of non-binary coded GAs [46, 47]. Many other genetic operators have been implemented according to different representations. For a more in-depth treatment of this subject, the reader is referred to [39].

2.2.2 Evolution Strategies

Although the PSO has been used vastly in this work as a vehicle to exhibit and validate the proposed models for incremental optimization, we did not restrict ourselves to this class of intelligent algorithms. ESs, for instance, is used to introduce

a hybrid implementation of the proposed incremental technique. In this section, we briefly introduce the ESs to provide necessary background for this later study. For a more thorough description of ESs, including a mathematical analysis of the convergence speed, a detailed comparison with other numerical optimization methods and Fortran implementations of a number of different variations, the reader is referred to [40].

ESs were developed by Rechenberg [51, 53] and Schwefel [48, 50]. In ESs, mutation is emphasized over recombination [29]. The two basic schemes of ESs are known as the *comma* strategy and the *plus* strategy. They are identified by the notation (μ, λ)-ES and $(\mu + \lambda)$-ES, respectively, where μ is the size of the parent population and λ is the number of offspring that are produced in a single generation before selection is applied. In a (μ, λ)-ES, μ of the λ offspring is selected to become the parent population in the next generation. Since the parents are totally replaced, this strategy may not be able to preserve good solutions from the previous generation. But it increases the diversity of the population. In a $(\mu + \lambda)$-ES, the μ parents and the λ offspring are merged into a single large population, on which selection is performed, picking the parent population of the next generation. This strategy preserves the best solutions discovered so far, so that the fitness of the best individual of the population is a monotonic function [30]. Different values of μ and λ could have large impact on the performance of ESs [30]. The original ES was a two-membered scheme consisting of one parent and one offspring [51], which can be denoted by (1+1)-ES under the definition given above. This (1+1)-ES is used in the study presented in Chapter 3 for its simple implementation, low cost and good performance when dealing with low dimensional problem as shown experimentally in [52]. Its procedure is described by the pseudo code shown in Figure 2.6.

Mutation, which is often the only evolutionary operator used in ESs, involves perturbing each element of the decision vector by an amount produced from a Gaussian distribution whose variance is adapted over time. Given the (1+1)-ES,

the *1/5 success rule* is used to adjust the variance, which can be described as follows [40]:

> From time to time during the optimum search, obtain the frequency of successes, i.e., the ratio of the number of successes to the total number of trials (mutations). If the ratio is greater than 1/5, increase the variance, if it is less than 1/5, decrease the variance.

This rule-of-thumb was developed by Rechenberg [53] as a result of his theoretical investigation of the (1+1)-ES, which has been shown to produce a high rate of convergence.

2.3 Particle Swarm Optimization

The PSO is a population-based optimization method proposed by Kennedy and Eberhart [44, 49]. It is famous for the ease of implementation, no requirement of gradient information and fast convergence. It can be applied to many problems in diverse fields of study, including hard function and combinatorial optimization, neural network design, planning and scheduling, industrial design, management

```
t = 0
Initialize P_t of 1 random individual from R^n
Evaluate fitness of individuals in P_t
WHILE (termination conditions is false)
{
        Clone the individual in P_t to produce 1 offspring
        Mutate the offspring to create variation
        Evaluate fitness of the offspring
        Select the better one from the parent and offspring individuals to produce P_t+1
        t = t+1
}
```

Legends: t is a discrete unit of time and P is a population.

Figure 2.6: Canonical(1+1)-ES

and economics, machine learning, and pattern recognition. However, it has a well-accepted flaw that the probability of premature convergence increases dramatically when the search space gets more complicated, which is often caused by the increase of dimensionality. Therefore, PSO is chosen as a vehicle to validate the macro models of incremental optimization proposed in this thesis. The basic definitions, implementation, convergence features and configurations of PSO are reviewed in this section.

2.3.1 Original PSO Algorithm

PSO is an evolutionary computation technique developed by Kennedy and Eberhart in 1995 [44]. The concept of particle swarm originated as a simulation of a simplified social system such as a bird flocking. The PSO algorithm maintains a population of randomly initialized solutions, where the population is called a swarm and each solution is named a particle. The particles "fly" through the search space to find the desired solution. Each particle i ($i = 1, 2, \ldots, s$), where s is the number of particles in the swarm, i.e. swarm size, possesses several characteristics that denoted by the following symbols:

- X_i: The current position of the particle;
- V_i: The current velocity of the particle;
- B_i: The personal best position of the particle.

Each particle keeps track of its coordinates in the search space, namely the personal best solution. The personal best position associated with particle i is the best position that the particle has visited, producing the best outcome for that particle. Considering a minimization problem, a position corresponding to a smaller objective value is regarded as producing a better outcome.

Another "best" that is tracked by the global version of the particle swarm optimizer is the overall best position, namely the global best solution. The global best solution of a swarm, denoted by $\widehat{B}$, is the position producing the best outcome obtained so far by any particle in that swarm.

At each time step, the PSO algorithm updates the velocity and the position for every particle in the swarm. The updating of velocity and position for a particle is conducted separately for each dimension $j \in [1, d]$. And $v_{i,j}$ and $x_{i,j}$ denote respectively the jth dimension of the velocity vector and the position vector associated with the ith particle. The updating rules are described by the following equations:

$$v_{i,j}(t+1) = v_{i,j}(t) + c_1 r_{1,j}(t)[b_{i,j}(t) - x_{i,j}(t)] + c_2 r_{2,j}(t)[\widehat{b}_j(t) - x_{i,j}(t)] \tag{2.8}$$

$$x_{i,j}(t+1) = x_{i,j}(t) + v_{i,j}(t+1) \tag{2.9}$$

In Equation (2.8), there are two independent random sequences, $r_1 \sim U(0,1)$ and $r_2 \sim U(0,1)$, used to effect the stochastic nature of the algorithm. The values of r_1 and r_2 are scaled by constants $0 \leq c_1, c_2 \leq 2$. These two constants are called acceleration constants, which influence the maximum size of the step that a particle can take in a single iteration.

Besides, the algorithm updates the personal best of each particle and the global best of the swarm at the end of each iteration. The update equations:

$$B_i(t+1) = \begin{cases} B_i(t), & \text{if } f(X_i(t+1)) \geq f(B_i(t)) \\ X_i(t+1), & \text{if } f(X_i(t+1)) < f(B_i(t)) \end{cases} \tag{2.10}$$

$$\begin{aligned} \widehat{B}(t) &\in \{B_1(t), B_2(t), \ldots, B_s(t)\} \\ f(\widehat{B}(t)) &= \min\{f(B_1(t)), f(B_2(t)), \ldots, f(B_s(t))\} \end{aligned} \tag{2.11}$$

The PSO algorithm employs repeated application of the updating equations presented above as follows:

1. Initialize a population of particles with random positions and velocities on d dimensions in the input space.

2. For each particle, evaluate the desired optimization fitness function in d variables.

3. Compare particle's fitness evaluation with particle's personal best (*pbest*). If current value is better than the pbest, then set pbest value equal to the current value and the pbest position equal to the current coordinates in d-dimensional space.

4. Compare fitness evaluation with the swarm's overall previous best. If current value is better than the global best (*gbest*), then reset gbest to the current particle's index and position.

5. Change the velocity and position of the particle according to Equation (2.8) and Equation (2.9), respectively.

6. Loop to step 2 until a stopping criterion is satisfied, usually a sufficiently good fitness or a maximum number of iterations.

Note that the velocity is clamped to the range $[-V_{max}, V_{max}]$ to prevent the algorithm from diverging. In other words, if the sum of accelerations causes the velocity on that dimension to exceed v_{max}, then the velocity on that dimension is limited to v_{max}. If the input space is defined by the bounds $[-X_{max}, X_{max}]$, then the values of V_{max} are typically set so that $V_{max} = k \cdot X_{max}$, where $0.1 \leq k \leq 1.0$ [54].

2.3.2 Convergence Conditions of the PSO

In [54–56], Bergh mathematically analyzed and experimentally illustrated the convergence characteristics of the PSO. Some of the conclusions are extensively used in this thesis for parameter setting and theoretical analysis. The useful statements are reproduced below without detailed proof. If reader are interested in those proofs, it is recommended to refer to Bergh's work.

As shown in Equation (2.8), the application of the two random sequences induces the stochastic feature. If these stochastic components are got rid of, the trajectory of a particle can be accurately described. According to Bergh, it is analogous to the dampened vibrations observed in a spring-dashpot system [54]. The convergence behavior is related to the relationship between the inertia weight (w) and the acceleration constants. To get convergent trajectories, the following condition must be satisfied:

$$w > \frac{1}{2}(c_1 + c_2) - 1. \tag{2.12}$$

Since it is customary to set $c_1 = c_2 = c$, Equation (2.12) can be simplified to:

$$w > c - 1. \tag{2.13}$$

With regard to the stochastic components, they influence the acceleration coefficients, making them fluctuate from zero to the value of the acceleration constants. Consequently, even if the relationship between the inertia weight and the acceleration constants does not satisfy Equation (2.13), the relationship between the inertia weight and the acceleration coefficients has chances to satisfy that convergence condition. It can be inferred from Equation (2.13) that, given a certain w value, the critical value of c is $w + 1$.

For convenience, let $\phi_1 = c_1 r_1 = c r_1$, $\phi_2 = c_2 r_2 = c r_2$ and $c_{crit} = w + 1$. Since

r_1 and r_2 are values from uniformly distributed sequences, the ratio of c_{crit} to c implies the probability of converging without disruptions along the trajectory. This important ratio is defined as:

$$\phi_{ratio} = \frac{c_{crit}}{c} = \frac{w+1}{c} \tag{2.14}$$

2.3.3 Parameter Settings for the PSO

An algorithm with a proper set of parameters should maintain a balance between exploration and exploitation. Exploration is the tendency of the algorithm to search new regions of the search space, while exploitation is the ability of conducting thorough search in vicinities of the current solutions.

Apparently, the value of w decides how much the moving direction of a particle relies on its previous velocity. That is why it is named the inertia weight. With small inertia, the particles would quickly aggregate to the global best solution found so far. Thus, smaller w values usually result in faster rates of convergence.

Nevertheless, the convergence rate would be offset by divergent steps. As mentioned above, the frequency of the divergent steps could be measured by the ϕ_{ratio}. When a w is given, the value of ϕ_{ratio} is decided by c. If the value of c is chosen to make $\phi_{ratio} \geq 1$, the algorithm will not have much explorative behavior and converges rapidly. For the same w, if c is set to a larger value to make ϕ_{ratio} slightly smaller than 1, the algorithm will posses more explorative behavior while looses some exploitative ability [31, 54].

Although the convergence of the PSO can be guaranteed by choosing proper parameter values, the point it converges to may not be a global or even a local optimizer. Therefore, some modifications are suggested to PSO in order to change it to be a local/global search algorithm.

2.4 Related Work to Incremental Models

Much of the work in the field of optimization, especially with intelligent algorithms, is related to the topics investigated in this thesis. A vast number of new algorithms can be constructed by modifying certain components of canonical intelligent algorithms or simply connecting different optimization algorithms in a proper manner. For example, a local search algorithm such as SA can be applied after PSO in the late stages of the optimization process to refine a solution, resulting in a global search. Replacing the SA with other local search method yields another new hybrid algorithm. Such algorithms can be organized into the context of Memetic Algorithms [58]. In this section, we mainly discuss the work most kin to our study, including the related modifications made to PSO for global optimization and well-known multi-objective PSO algorithms.

2.4.1 Challenges and Solutions on Global Optimization

Although the biologically inspired intelligent algorithms are effective in solving some global optimization problems which are difficult for classical numerical methods, some features of objective function, such as ridges and local optima, often obstruct them from converging to the global optimum. The algorithmic challenge in handling ridges is to change multiple variables simultaneously in order to search in the direction of the ridge orientation and thereby avoid reduction in fitness. A lot of problems with ridges could be successfully solved by self-adaptive ES [40]. However, the self-adaptive ES is not suitable for high-dimensional problems. According to [30], a chromosome of self-adaptive ES should include the "object parameter" as well as the "strategy parameters" and the "rotation parameter". For instance, a 100-dimensional problem requires 4950 angles to allow rotation between all dimensions [41]. The problem with local optima, namely multi-modal problem, is

also quite common and unavoidable in function optimization problems. It is well perceived to be hard to handle, especially when the number of local optima is large. The biologically inspired canonical intelligent algorithms have a tendency to stagnate on such local optimum because escaping from them may require a significant amount of backtracking, namely "downhill movement", before new fitness improvements occur. Thus, a great deal of work has been dedicated to improving those algorithms. Our incremental technique is one of them.

The idea of building incremental models can be roughly explained by a scenario in the daily life. If it is known there is a moth-eaten hole in an apple, how can it be found? A person may peel the apple layer by layer until the hole is found. But it would take a lot of effort to reach the hole if it is deeply inside. Another person may mesh the apple into small pieces to find the one with the hole. Besides the meshing process, it would be inefficient to check the small pieces one by one. So, mostly the apple is sliced from different directions. Since the flesh will turn brown around a moth-eaten hole, once any clue is found, cut around it to find the direction along which the brown color turns darker. Along this direction, the hole would be found soon. Considering a global optimization problem, it is just like an apple with a moth-eaten hole, i.e. its global optimum. Thus, seeking for the global optimum is analogical to searching the hole in the apple. Inspired by the searching approach intuitively used in our life as described above, a cutting plane mechanism is proposed for global optimization. Based on this mechanism, the incremental technique was designed. The cutting planes are obtained by fixing all the variables but one. If the number of the fixed variables decreases, cutting hyper-planes are obtained. The incremental technique reduces the dimension for searching by cutting the search space into cutting planes/hyper-planes. It consists of a series of phases and one more variable is fine tuned in each phase. This means that the number of phases is equal to the number of variables. Each phase is composed of two stages, one is called single-variable optimization (SVO) and the other one is called

multi-variable optimization (MVO). The cutting plane mechanism is used in both SVO and MVO. In SVOs, fine tuning on a single variable is conducted on several cutting planes, and the cutting planes are adjusted according to the results of the fine tuning taken on them. In MVOs, the fine tuning on the incremented variable set is conducted on several cutting hyper-planes, and the cutting hyper-planes are adjusted according to the results of the fine tuning. Additionally, an integration process is designed for creating initial population of MVO. The integration is taken on the results obtained by the SVO in the current phase and the MVO in the last phase, and the generated individuals constitute part of the initial population of the MVO in the current phase. In some other fields, the "incremental" concept has been already proved feasible and effective, including incremental learning in the area of machine learning [27] and incremental evolution for multi-objective problems [74].

With regard to the implementation of the components of the proposed incremental technique, including the fine tuning on cutting planes/hyper-planes and the adjustment of cutting planes/hyper-planes, it is not restricted to any specific algorithm. In this thesis, PSO is employed as a main vehicle to apply the incremental model. This is because it is commonly accepted that the standard PSO does not scale well to high-dimensional problems without dramatically increasing the swarm size, as it lacks an efficient means for maintaining genetic diversity [26, 54]. An incremental technique may be a good choice for ensuring the genetic diversity because SVOs are conducted independently and new individuals are integrated into the population of MVO in each phase. Firstly, PSO is used both for fine tuning and for adjusting cutting planes/hyper-planes, resulting in an algorithm called Incremental Particle Swarm Optimization (IPSO). Further, (1+1)-ES is employed to investigate a hybrid implementation of the incremental model for its prominent ability of local search. It replaces PSO in fine tuning, which results in an algorithm called Particle Assisted Incremental Evolution Strategy (PIES).

In the context of PSO, various modifications to the standard PSO have been proposed. Eberhart *et al.* modified PSO to an asynchronous version [75]. This asynchronous version of PSO (ASPSO) differs from the standard version in that the global best is updated after updating each particle position. Eberhart, who is one of the creators of the standard PSO, argued that the rate of convergence of the standard PSO would be improved by this asynchronous updating. Blackwell and Bentley introduced a repulsive force to the position updating equation [76]. Their algorithm (CSPSO) is based on an analogy of electrostatic energy with charged particles. They assume there are some particles in the swarm, which repel each other. The repulsive force has an identical form to the familiar electrostatic inverse square law between charged particles. But they only applied CSPSO to a dynamic problem. How the algorithm works on static global optimization is yet to be investigated. Besides, the calculation of the repulsive force would result in a rather high computational cost. Løvbjerg et al. combines the reproduction and recombination of GAs, as well as the subpopulation mechanism often used to improve GAs, with the standard PSO in order to achieve a faster convergence and a higher success rate [77]. Their algorithm (HPSO_BS) still treats the search space as a whole. A potential drawback of adjusting the decision variables all together is the premature convergence, as it is possible that some of the variables may move away from their optimal values while some others approach their optimal values. Therefore, IPSO/PIES takes fine tuning on the decision variables one after another under a scenario of continuous incremental optimization.

A significant characteristic of the proposed incremental model for global optimization is the cooperation of global and local search. The adjustment of cutting planes/hyper-planes globally guides the search, while the fine tuning on the cutting planes/hyper-planes does the job of local search. The benefits of cooperation between global and local search have been studied within the memetic algorithm (MA) literature [78]. MAs combine a population-based global search and the heuristic lo-

cal search made by each individual. Some researchers in the MA field have tried to combine the power of EAs and PSO. In [79], a memetic learning algorithm, SMNE, was introduced for developing neurocontrollers. For SMNE, a symbiotic GA is employed to perform a global explorative search, while PSO is used to perform local exploitive search after each EA generation. Also in [24], a hybrid algorithm of GA and PSO was proposed while the roles are exchanged. PSO performs global search, while GA is used to refine the solutions obtained by the PSO. The results showed that this hybrid algorithm outperforms simple PSO and simple GA. However, both these EA-PSO hybrids are just a simple combination of two algorithms, which is done by taking the population of the global search either after each generation or when the improvement starts to level off and using it as the starting population of the local search. In contrast, global search and local search in the incremental technique take place in different dimensions as the cutting plane mechanism is used. In other words, the refinement in IPSO/PIES is instead taken dimension by dimension, like sieving the solution space by sieves with higher and higher dimensionality.

Moreover, parallel implementation of the PSO-based incremental optimization is studied in order to improve its efficiency, resulting in a parallel IPSO (PIPSO). This parallel mechanism enhances the information sharing, which can find its root in the "blackboard" model of cooperation optimization. There are two categories of cooperation optimization models. One of them distributes the complete individual solutions over a set of subpopulations, while the other one distributes the subcomponents of individual solutions over a set of subpopulations [54].

The representative of the former one is the island model proposed by Grosso [59]. This model is used to partition a large population into several smaller subpopulations, so that each subpopulation can be evolved separately. Periodically some individuals "migrate" from one subpopulation to another, allowing the subpopulations to share information regarding the solutions discovered so far. It aims to

preserve diversity in order to prevent premature convergence. But how and when the migration should take place is a hard-to-decide issue, with many parameters to be decided.

Many researchers have contributed in the latter category of cooperation optimization. Potter [45,60] proposed Cooperative Coevolutionary GA (CCGA), which decomposes the solutions of a d-dimensional problem into d subpopulations, each subpopulation corresponding to one of the variables. Potential solutions to the problem are constructed by taking one representative from each of the d subpopulations to build a d-dimensional vector. In CCGA, although different subpopulations are cooperating to find the solution, communication among subpopulations is limited with only one complete solution being constructed. Bergh investigated the application of Potter's model on PSO, resulting in a novel algorithm called cooperative PSO (CPSO) [54]. Similarly, the CPSO partitions the d-dimensional decision vectors into d swarms of one-dimensional vectors, with each swarm representing a dimension of the original problem. Each swarm attempts to optimize a single component of the solution vector, essentially a one-dimensional optimization problem. A *context vector* is used to provide a suitable context in which the individuals can be evaluated. This vector is constructed by taking the global best particle from each of the d swarms and concatenating them together. To calculate the objective value for all particles in swarm i, the other $d-1$ components in the context vector are kept constant while the ith component of the context vector is replaced in turn by each particle from the ith swarm. Since the CPSO is in CCGA-style, it also inherits the communication flaw of CCGA, so that it tends to be trapped when the variables of a problem are highly correlated. Clearwater *et al.* [57] investigated the behavior of cooperating agents in the context of constraint-satisfaction problems. The "blackboard" model they proposed uses a shared memory, which each individual can post hints to or read hints from. In this case, global communication is available. The authors observed a super-linear speedup when their agents were co-

operating via the blackboard, while a linear speedup when multiple non-interacting agents were used.

2.4.2 Issues on Multi-Objective Optimization Algorithms

In the field of multi-objective optimization, multi-objective GAs (MOGAs) have been widely studied and numerous MOGAs have been suggested [35, 36, 94–101, 108]. Since MOGAs are not the focus of this study, they are not reviewed in detail. However, SPEA [30, 31] and NSGA-II [28] are two representatives of the MOGAs which attract most attention, and is used for comparison in the later study of this thesis. The main features of these two state-of-the-art MOGAs are introduced here. SPEA maintains an external population to store the non-dominated solutions discovered so far. This external population participates in all genetic operations. The dominated solutions in the population obtained from combining the external and current population are assigned a fitness which is equal to the sum of the fitness values of their dominating solutions. Such a fitness assignment scheme ensures that the non-dominated solutions always have better fitness values than those being dominated, therefore drive the evolution towards the Pareto-optimal front. In addition, a deterministic clustering technique is used to select the most representative individuals if the size of the external population exceeds a pre-set threshold. A significant difference that distinguishes NSGA-II from SPEA is that no external archive is used to store non-dominated solutions. In NSGA-II, the crossover and mutation are performed to generate as many offspring as the parent population in every generation. Then the whole population is sorted based on the non-domination and each solution is assigned a fitness value equal to its non-domination level. The fitter half of the population survives. If it is necessary to select solutions belonging to the same level, the solutions living in sparser region have a greater chance to survive.

Given the competitive performance of PSO on global optimization, researchers tend to extend it to the field of multi-objective optimization. Moore and Chapman [61] emphasized the significance of performing both an individual search that is called a cognitive component, and a group search that is called a social component. However, no diversity maintenance scheme is adopted in their algorithm. Ray and Liew [62] combined some concepts of EAs with the particle swarm. But they mainly focused on handling MOPs with constraints, by using a multi-level sieve. More recently, external archive has been used by some researchers to realize elitism for PSO-based multi-objective optimization. Fieldsend and Singh [63] proposed an approach in which they used an unconstrained elite archive. This archive adopts a special data structure called "dominated tree" to store the non-dominated individuals found along the search process, and interacts with the primary population in order to define local guides. In addition, the approach uses a "turbulence" operator that is basically a mutation operator that acts on the velocity value used by PSO. However, it seems to have problems when handling multi-frontal MOPs. Mostaghim and Teich [64] proposed a sigma method in which the best local guides for each particle are adopted to improve the convergence and diversity of a PSO-based multi-objective optimization. They also used a mutation operator, but applied it on decision variable space. The idea of the sigma approach is similar to compromise programming [65]. The use of the sigma values increases the selection pressure of PSO that is already high, which may cause premature convergence. Li [66] proposed an approach which adopts the main mechanisms of the NSGA-II [28] in a PSO-based multi-objective optimization algorithm. This approach showed a very competitive performance with respect to the NSGA-II. Carlos and Coello [10] adopted an external archive similar to the adaptive grid of PAES [67] and used mutation operator both on the particles and on the range of each decision variable. Their approach exhibits high efficiency in solving various MOPs, therefore is chosen to be the vehicle to which the incremental model

proposed in this study is applied.

What should be noted is that almost all of them treat the objectives of an MOP as a whole and evolve them together. Hu and Eberhart [68,69] proposed a dynamic neighborhood PSO for multi-objective optimization. In this algorithm, only one objective is optimized at a time using a scheme similar to Lexicographic ordering [65]. However, Lexicographic ordering tends to be useful only when few objective functions are used (two or three), and it may be sensitive to the ordering of the objectives. Inspired by a powerful MOGA named incremental MOGA (IMOGA) [85], we considered building a macro incremental model for multi-objective optimization and applying it to MOPSO. The superiority of IMOGA over other MOGAs has been shown in [85], where it outperformed three well-known MOGAs, NSGA-II [35], SPEA [94] and PAES [96], on all the problems tested. This incremental model evolves the objectives one by one and benefit from inheritance as in IMOGA. Its most important characteristic is the compatibility to most of the intelligent algorithms, rather than only MOGA or MOPSO.

The development of the incremental multi-objective optimization arose from an idea that performance of a certain tool improves as its task gets easier. It is assumed that the performance of a method for multi-objective optimization improves, or at least would not degrade as the objective set gets smaller, and the Pareto-optimal points are likely to remain Pareto-optimal after objective increment. Applying this rationale, if the initial solutions from the first few objectives contain better candidates, upon subsequent objective increment, they are most likely to stay, and also improve the accuracy and quality of the solutions. Thus, an incremental optimization can be more efficient. The incremental model can also be applied to the mentioned state-of-art MOGAs, NSGA-II, SPEA and PAES, resulting in INSGA-II, ISPEA and IPAES. Performance of the obtained incremental algorithms has been shown better than the original ones [110].

A significant issue of the incremental model, the ordering of objectives, is first addressed in this thesis. The IMOGA is used as a vehicle for this study. With regard to the original IMOGA, the objectives were handled according to the original order they were given. Whether this is a good choice or not is questionable. According to our study, the performance of IMOGA fluctuates a lot as the objective order changes. Generally the original objective order does not result in the best performance.

In this thesis, why objective ordering has influence on the performance of incremental MOP solving is analyzed. Based on the analysis, two ordering approaches are designed. Experiments are then conducted to compare and verify these approaches. In addition, we explain why the two ranking metrics are reasonable and why one of the approaches is recommended.

Some researchers have studied the relationship among objectives within the context of evolutionary multi-objective optimization [111, 112]. Purshouse and Fleming categorized the relationship into conflicting, harmony and independent, and analyzed the effect of each category on evolutionary multi-objective optimization (EMO) [111]. Their analysis is consistent with the assumption made in this paper that MOPs with more conflicting objectives would be harder to solve. Schroder defined the level of conflict as a weighted-sum of the crossings between pairs of objective regions [112]. But this method requires additional preference information to partition each objective range into a number of regions. In this thesis, comparison-based methods to rank objective pairs and individual objectives are proposed for objective ordering. The major idea is that the conflict level of each objective pair can be compared by the range of their Pareto fronts, and the difficulty in handling each individual objective can be compared by the performance of the optimization algorithm solving them. With the comparison results, the objective pairs can be ranked by their ranges (of the Pareto-optimal fronts) and the individual objectives can be ranked by their performance (of the optimization algorithm solving them).

After ranking, we shall evolve as early as possible the objective pairs with smaller ranges and/or the objectives with better performance. An ordering strategy is drawn comprising both metrics. The results show that the range metric plays a major role in ordering while the objective performance metric plays a minor role. Based on this strategy, the found best objective order would be consistent with the optimal order obtained by exhaustive searching experiments.

So far, the background knowledge on CIAs-based optimization and the research works related to this study have been discussed. The following chapters are dedicated to the study on incremental optimization both in input and output spaces.

Chapter 3

Incremental Global Optimization

This chapter studies incremental optimization in the input space. The basic idea here is to transform a global optimization problem to a series of sub-problems. The dimensionality of these sub-problems increments from low to high, where the highest one is equal to the dimensionality of the original problem. To implement incremental global optimization, a cutting plane mechanism is proposed. Based on this mechanism, an incremental model in the input space is built. PSO is employed as a vehicle to apply the incremental model, resulting in an algorithm called Incremental Particle Swarm Optimization (IPSO). Furthermore, (1+1)-ES is employed to investigate a hybrid implementation of the incremental model, resulting in an algorithm called the Particle assisted Incremental Evolution Strategy (PIES). The performance of IPSO and PIES is compared with improved PSO algorithms to study the efficacy of the incremental model on several benchmark global optimization problems.

3.1 Orthographic Projection of the Search Space

3.1.1 Motivation

As we know, a 3-dimensional object can be described exactly by three-view orthographic projection drawing, which is some kind of mechanical drawing. A three-view orthographic projection drawing shows the front, top, and right sides of an object as shown in Figure 3.1. An important factor in a three-view drawing is the relationship between height, width, and depth. The top and front views share the width of the object, the top and side views share the depth, and the front and side views share the height.

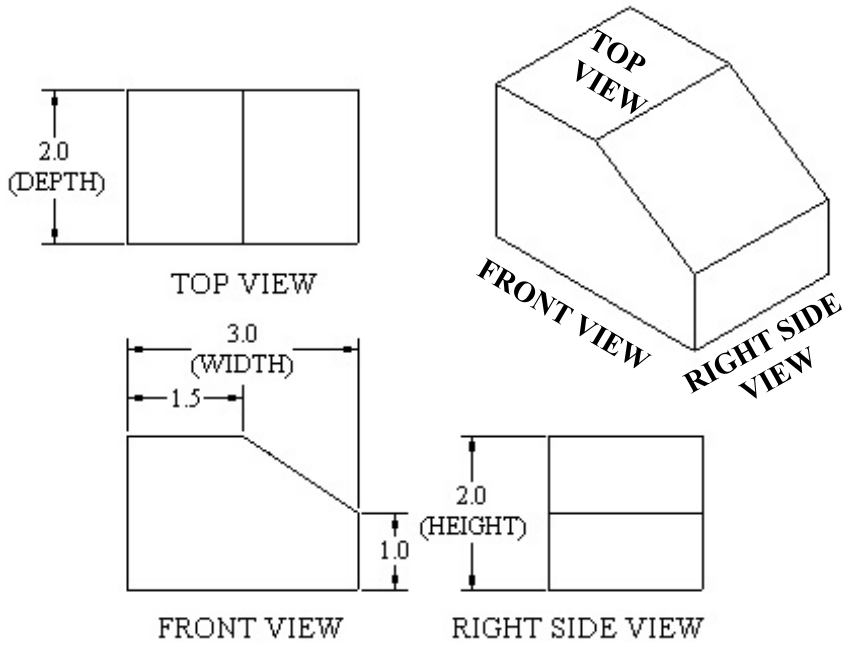


Figure 3.1: A three-view orthographic projection

With respect to a global optimization problem, the aim is to find the optimal value of the objective function. If we visualize an objective function with d variables as a hyper-surface in the $(d+1)$-dimensional space, global optimization requires

to find the nadir (or the zenith in maximization problem) of the hyper-surface. An important observation on the three-view orthographic projection drawing is that the height information will not be lost with orthographic projection from the front view or the side view. Inspired by this observation, we consider taking orthographic projection of objective function from "variable view", which means to orthographically project the corresponding hyper-surface onto variable-objective value planes. The detailed mathematic description of this concept is presented as follows.

3.1.2 Effect of orthographic projection

Consider a single objective minimization problem with d attributes in the input space. We can formulate the optimization problem as finding $\mathbf{x} = [x_1, x_2, \ldots, x_d]$ to minimize the value of $y = f(\mathbf{x})$ within the feasible input region S.

For the ease of description, some definitions are given as follows:

1. Feasible input region S is the set of all vectors that satisfy the constraints and bounds of the problem.

2. $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_d, \mathbf{u}_{d+1}\}$ are the orthogonal bases in the $(d+1)$-dimensional space $\mathbb{R}^{d+1}$, corresponding to $\{x_1, x_2, \ldots, x_d, y\}$. A 3-dimensional example is demonstrated in Figure 3.2.

3. Orthographic projection refers to the projection along the orthogonal bases, which means to be projected onto a unit vector $\mathbf{u}_i$, $i = 1, 2, \ldots, d+1$.

4. P_{x_i-y}, $i = 1, 2, \ldots, d$, is the boundary of the orthographic projection of the original function $y = f(\mathbf{x})$ on the $x_i - y$ plane. We use a function $y^{(i)} = f^{(i)}(x_i)$ to describe P_{x_i-y}.

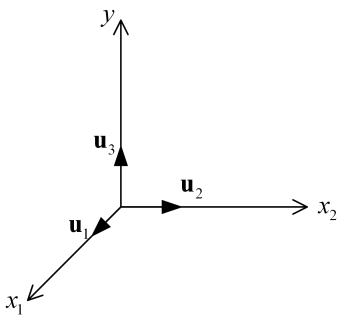


Figure 3.2: Orthogonal bases in $\mathbb{R}^3$

5. To facilitate discussion without losing generality, assume that there is only one global optimal solution $y^g = f(x_1^g, x_2^g, \ldots, x_d^g)$ for the original problem. Thus one optimal solution $(x_i^*, y^{(i)^*})$ for each projected problem could be found.

For incremental optimization, the original problem can be projected into d one-dimensional sub-problems. The optima of the sub-problems are denoted as: $\{(x_1^*, y^{(1)^*}), (x_2^*, y^{(2)^*}), \ldots, (x_d^*, y^{(d)^*})\}$.

Theorem 3.1: *The minimum of P_{x_i-y}, $(x_i^*, y^{(i)^*})$, is the projection of the global minimum $(x_1^g, x_2^g, \ldots, x_d^g, y^g)$ of the original problem on the $x_i - y$ plane, $i = 1, 2, \ldots, d$.*

Apagoge is used to prove this statement as follows:

Assume $(x_i^*, y^{(i)^*})$ is not the projection of $(x_1^g, x_2^g, \ldots, x_d^g, y^g)$ on the $x_i - y$ plane, $i = 1, 2, \ldots, d$.

When we project $y = f(\mathbf{x})$ onto the $x_i - y$ plane, each point $y = f(x_1^o, x_2^o, \ldots, x_d^o, y^o)$ on the original hyper-surface should correspond to a point in the area of the orthographic projection. When taking orthographic projection along all the directions except $\mathbf{u}_i$ and $\mathbf{u}_{d+1}$ in the space $\mathbb{R}^{d+1}$, there is $|\mathbf{u}_j \times \mathbf{u}_j| = 0$, $j = 1, 2, \ldots, d$ and $j \neq i$. Since there is $\mathbf{u}_i \cdot \mathbf{u}_i = 1$ and $\mathbf{u}_{d+1} \cdot \mathbf{u}_{d+1} = 1$ in the $x_i - y$ plane, the

projection point $(x_i^p, y^{(i),p})$ will be

$$\begin{cases} x_i^p = x_i^o \\ y^{(i),p} = y^o \end{cases}.$$

That is to say, the "height" information of the ith and the $d+1$th dimensions is retained, and that of the other dimensions is discarded.

So there is at least one point on the original hyper-surface corresponding to each point $(x_i, y^{(i)})$ on P_{x_i-y}, a curve described by $y^{(i)} = f^i(x_i)$.

According to the assumption, there should be a point $(x_i^c, y^{(i),c})$ in the area of the projection in the $x_i - y$ plane other than $(x_i^*, y^{(i)^*})$ corresponding to the global minimum $(x_1^g, x_2^g, \ldots, x_d^g, y^g)$, which has

$$\begin{cases} x_i^c = x_i^g \\ y^{(i),c} = y^g \end{cases}.$$

With regard to the global minimum, $y^g = \min\{y^o\}$, then it can be deduced that $y^{(i),c} = \min\{y^{(i),p}\}$, that is to say $y^{(i),c} < y^{(i)^*}$. However, there is the premise that $(x_i^*, y^{(i)^*})$ is the minimum point of P_{x_i-y}, the boundary of projection on the plane, which means this point is the minimum in the area of projection. Obviously, the conclusion contradicts with the premise, and it is proved that $(x_i^*, y^{(i)^*})$ is the projection of the global minimum point on the $x_i - y$ plane.

In some cases the assumption that there is only one global optimal solution of the original problem may not hold. This means there could be multiple global optimal solutions of the original problem. Nonetheless, it is obvious that the proof above still holds in the sense that the global optimal solutions will not lose their predominance in terms of y value after taking orthographic projection. The only difference could be that multiple optimal solutions would be found in some or all

of the projected problems, which are the projections of the original global optimal solutions. Obviously, the conclusion can be generalized to orthographic projection of a higher dimension. The generalized conclusion should be that, the minima of the boundary function, namely the projection of the hyper-surface corresponding to the original objective function, are the projection of the global optima.

3.2 Cutting Plane Mechanism

In Section 3.1, it has been proved that if the exact boundary functions of the orthographic projection can be obtained, the global optimum can be easily found by solving a one-variable problem. Unfortunately, usually we cannot find the exact function which describes the projection boundaries. Nevertheless, the features and concepts discussed above can be still made use of by employing a cutting plane mechanism.

For the ease of description, some definitions are given as follows:

1. For an optimization problem with d variables, we need to fix some variables and evolve the other variables. The fixed ones are called *unconcerned variables*, denoted by $\mathbf{x}^{uc}$, while the rest are called *concerned variables*, denoted by $\mathbf{x}^{c}$.

2. A point in the space of concerned variables is called the *projection* of the corresponding point in the original space and that the original point is the *pre-image* of the projection.

3. When there is only one concerned variable, the projection method is called *cutting plane mechanism*. The concerned variable-objective value plane is the *cutting plane*, and the cutting plane intersects the original hyper-surface resulting in an *intercepted curve*.

4. If the fixed values for those unconcerned variables are equal to the values of corresponding variables of the global optimum, i.e. $|x_i^{uc} - x_i^g| = 0$(where $x_i^{uc} \in \mathbf{x}^{uc}$ and x_i^g is the value of corresponding variable of the global optimum), the cutting plane is called the *optimal cutting plane* (OCP for short). The cutting planes falling into the ε-region of an OCP, $|x_i^{uc} - x_i^g| \leq \varepsilon$ (ε is the tolerance), are called the *ideal cutting planes* (ICPs).

The cutting plane mechanism could reduce the problem to a one-variable problem (e.g. the concerned variable is x_1) and this variable will be finely tuned. To form a cutting plane, the unconcerned variables $(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_d)$ are treated as dummies by setting them at certain fixed values, which means the cutting plane is applied to intersect the hyper-surface. Considering a two-variable problem $y = f(x_1, x_2)$ as an example, with the assumption that x_1 is the current concerned variable and the unconcerned variable x_2 is treated as a dummy by setting $x_2 = a$, the cutting plane is the gray area in Figure 3.3 and the intercepted curve in the surface is shown in Figure 3.4.

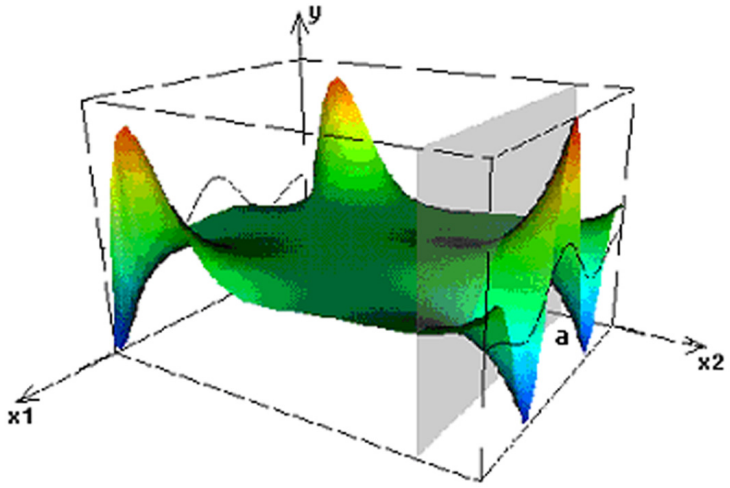


Figure 3.3: Cutting plane for a two-variable problem

As shown in Figure 3.3, $P(x_{P,1}, y_P)$ is the optimum point of the intercepted curve. Obviously, only if the cutting plane is the OCP or an ICP, $P'(x_{P,1}, a, y_P)$, the pre-image of P will be the desired global optimum or a satisfactory solution

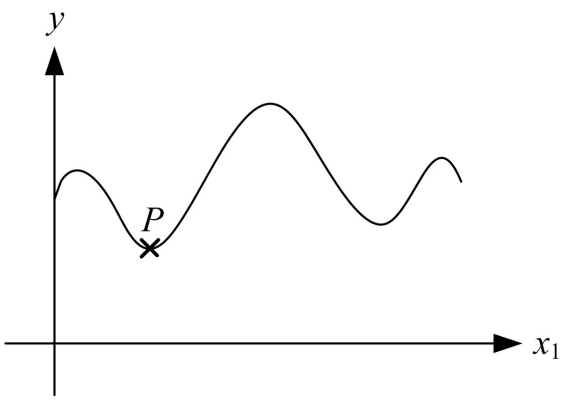


Figure 3.4: Intercepted curve in the surface

very close to the global optimum. Although the cutting plane in the example is not the OCP and thus P is not the projection of the global optimum, this point may be a "good" solution because at least it is better than all the other solutions on the cutting plane. In other words, the objective value of the solution found in a cutting plane more or less indicates how promising the landscape around this cutting plane is. So the cutting planes could be adjusted according to the objective values of their optima. The optimum of a cutting plane is the optimum of the intercepted curve on it, which is obtained by the fine tuning on the corresponding concerned variable.

In view of the proof in Section 3.1 and the above discussion, the rationale of the incremental model can be stated as follows:

1. The global optimum of a projected function in any dimension is the projection of the original global optimum in that dimension. This conclusion implies the feasibility of lowering down the searching dimensionality.

2. Considering the cutting plane mechanism, the closer to the OCP a cutting plane is, the more significant its optimum is. This analysis suggests adjusting the cutting planes/hyper-planes according to the quality of their solutions.

3.3 Incremental Model in the Input Space

Please note that the "incremental model" in this chapter means the incremental model in the input space.

Undoubtedly, searching in a lower dimensional space is easier and less time consuming than in a higher dimensional space. So, the search can start from only one concerned variable and approach the ultimate global optima step by step, resulting in an incremental approach. This approach makes sense only if the information obtained before a concerned-variable increment can help the searching after the increment. The analysis in Section 3.2 shows the feasibility of such an incremental approach. The detailed implementation of the proposed incremental model is described below.

In the incremental model, the whole optimization process is divided into several phases. In each phase, one more variable is finely tuned. Among all the phases, the first phase is called initial phase, the last phase is called ending phase, and those in between are called intermediate phases. Each phase is composed of two stages. Firstly, searching is taken on some cutting planes and these cutting planes move according to the quality of the solution found on them. This stage is called SVO (single variable optimization). Next, the better-performing individuals obtained from stage one and the population obtained from the last phase are joined together to generate an initial population for MVO (multi-variable optimization). The procedure is shown in Figure 3.5, and the pseudo-code of the incremental model is given below in Figure 3.6.

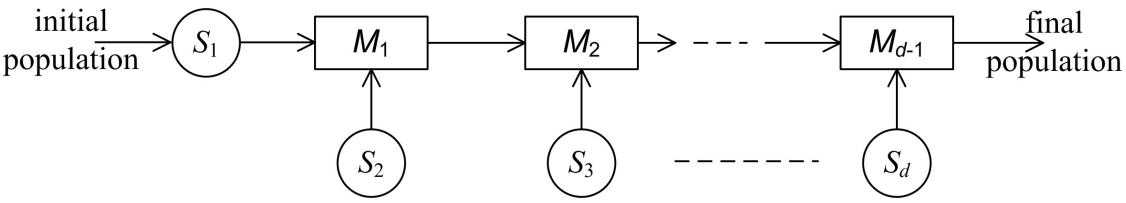


Figure 3.5: Incremental Model in the input space

Here, S_i $(i = 1, 2, \ldots, d)$ stands for SVO on variable i, and M_j stands for MVO with regard to from variable 1 to variable $j + 1$, $j = 1, 2, \ldots, d - 1$, where d is the dimensionality of the original problem.

```
Randomly initialize a population;
Perform SVO with the first variable concerned;
Select s individuals to form the first multiple concerned variable population
(MVP);
Set k = 1;
Until (ALL THE VARIABLE HAVE BEEN OPTIMIZED) Do
{
        k = k + 1;
        Randomly initialize a population;
        Perform SVO on the population with the k th variable concerned;
        Select s individuals to form the k th single concerned variable population
(SVP);
        Integrate the k th SVP with the (k − 1) th MVP to form a new population;
        Perform MVO with the first k variables concerned;
        Select the best m individuals to form the k th MVP;
}
```

Figure 3.6: Pseudo-code of the incremental model in the input space

3.4 PSO-based Incremental Optimization in the Input Space

There is no restriction on the algorithms that are used to implement the incremental model. In this thesis, PSO is firstly used as a vehicle to apply the incremental model in order to overcome its disadvantage of poor scalability. When it is used for both the fine tuning and the cutting plane/hyper-plane adjustment, the resulting algorithm is called incremental particle swarm optimization (IPSO). Further, (1+1)-ES is used for a hybrid implementation of the incremental model, which

replaces the PSO in the fine tuning. It is chosen for its prominent local search ability. The resulting hybrid algorithm is called the particle assisted incremental evolution strategy (PIES).

3.4.1 Flaw of PSO

As stated in Chapter 2, the rationale of PSO is to make some outstanding positions found by a swarm so far act as leaders to determine the flying directions of the particles in the swarm. Since the input space is dealt with as a whole, the values of all the variables of any leader are considered good and used to adjust the variable values of the other particles accordingly. In fact, not all the variables of a leader are necessarily closer to the global optimum than those of a common particle. Thus, some movement may drive particles away from some promising positions. Please see the example below.

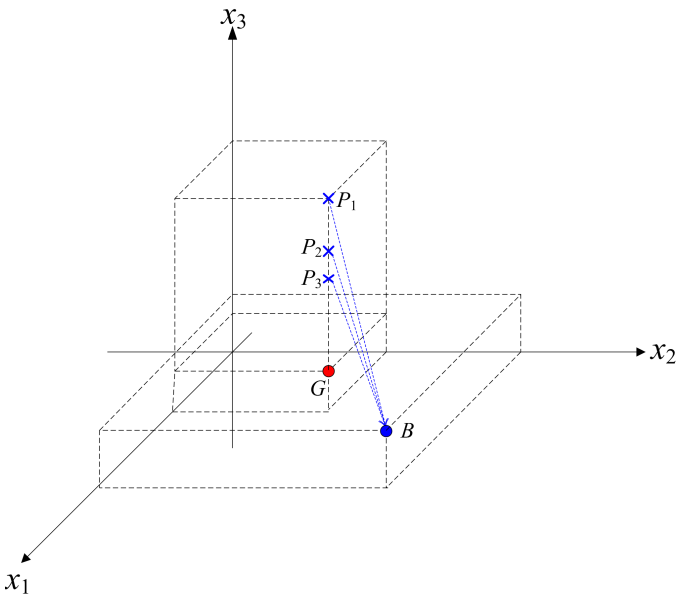


Figure 3.7: Degradation movement of particles

In Figure 3.7, P_1, P_2 and P_3 represent three particles that constitute a swarm, B represents the current global best position found by the swarm and G represents

the true global optimum. Since the objective values of these points, denoted by $f(\cdot)$, satisfy the following inequalities.

$$f(G) < f(B) \tag{3.1}$$

$$f(B) < f(P_1), f(B) < f(P_2), f(B) < f(P_3) \tag{3.2}$$

As shown in Figure 3.7, the location relationship among the points can be described as follows:

$$x_1^{P_1} = x_1^{P_2} = x_1^{P_3} = x_1^G < x_1^B \tag{3.3}$$

$$x_2^{P_1} = x_2^{P_2} = x_2^{P_3} = x_2^G < x_2^B \tag{3.4}$$

$$x_3^{P_1} > x_3^{P_2} > x_3^{P_3} > x_3^B > x_1^G \tag{3.5}$$

Assume that the current positions of P_1, P_2 and P_3 are their personal best positions and their previous velocities get close to zero. In this case, all the particles will move towards the global best position. Although the movement produces progress along x_3, it results in degradation along both x_1 and x_2. In other words, the particles are driven away from the true optimum along two dimensions while approaching it along only one dimension. If the objective values of the particles keep decreasing on their way flying to B, they may finally prematurely and incorrectly converge to B. Particularly, when the number of variables gets larger, namely the input space gets bigger, the phenomena described above would get more likely to happen, as it may be harder to guarantee progress along all the dimensions. That is to say, progress along some dimensions could be offset by degradation along some other dimensions, which may cause premature and incorrect convergence.

In order to avoid the "offsetting" effect discussed above, we apply the proposed incremental model to PSO, so that the global optimum can be approached

dimension by dimension, resulting in a novel PSO-based incremental algorithm, IPSO.

3.4.2 Procedure of IPSO/PIES

The IPSO/PIES works as follows (Assume there are d variables and N is the initial population size):

1. Set $k = 1$, where k is the phase number. Generate a population and implement SVO with regard to the first concerned variable. After that, m fittest individuals survive into MP_1 (MP_k represents the population with multiple concerned variables for phase k). Phase 1, namely the initial phase, then ends.

2. Set $k = k + 1$. The next phase starts.

3. Generate a population and implement SVO with regard to the kth concerned variable. After that, the m fittest individuals survive into SP_k (SP_k represents the population with single concerned variable for phase k).

4. Generate the initial population for the MVO in phase k, I_k, which is the result of integration operation on SP_k and MP_k.

5. If the size of I_k is larger than N, select the N fittest individuals. Then perform MVO on I_k on the concerned variables which including the first k variables. After searching, the evolved population MP_k is obtained. Phase k ends.

6. If the stopping criterion is not reached, go to step 2. Otherwise, the incremental optimization process finishes. The fittest individual in the final population is the obtained solution.

3.4.3 Components of SVO and MVO

SVO aims to evolve the population with regard to only one concerned variable, while optimization in MVO involves more than one concerned variables. Both SVO and MVO comprise of the following two components:

Fine tuning on cutting planes/hyper-planes

In SVO, fine tuning is taken on the each cutting plane, which only involves one concerned variable, which means the standard PSO (in IPSO) or the standard (1+1)-ES (in PIES) is conducted only on that variable. Thus, searching on a certain cutting plane is in equivalence to solving a one dimensional problem by PSO.

For each cutting plane, a random swarm will be generated to optimize the intercepted curve on it. Every particle in this swarm comprises of only one component, a value for the concerned variable. And they will fly along the dimension of that variable. Please note that the objective value of a particle will be evaluated by filling its value as well as the values of the unconcerned variables of the cutting plane into the objective function.

In Figure 3.8, the concerned variable-y plane is a cutting plane and the curve is the intercepted curve at the original hyper-surface intercepted by the cutting plane. In this cutting plane, all the particles have the same values for the unconcerned variables. The difference of their objective values is only decided by the different values of the concerned variable. The aim of searching the cutting plane is to find the global optimum on the intercepted curve.

In contrast, the fine tuning in MVOs is taken on the cutting hyper-planes. It involves more than one concerned variables, which equal to the sequence number

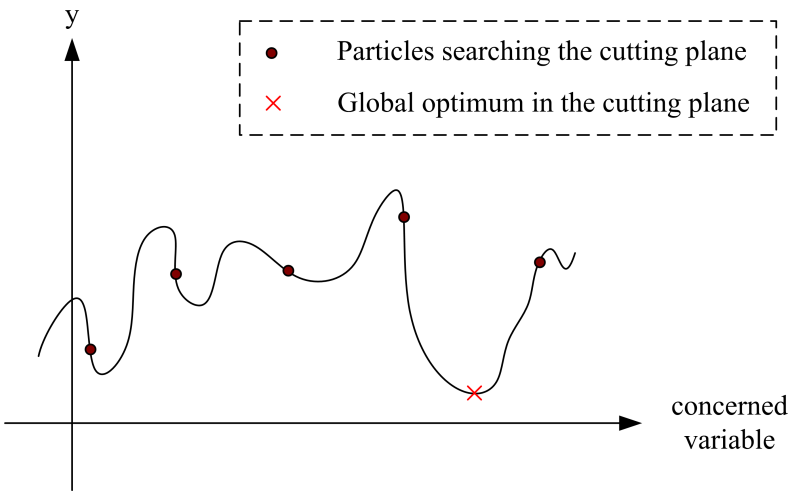


Figure 3.8: Illustration of searching on a cutting plane

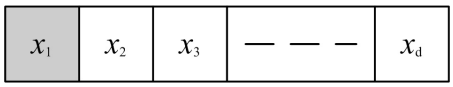
of current phase. In other words, we search the projection of global optima in cutting spaces with continuously incremented dimensions rather than in a cutting plane. Thus, the standard PSO (in IPSO) or the standard (1+1)-ES (in PIES) is conducted on the whole set of those concerned variables.

Adjustment of cutting planes/hyper-planes

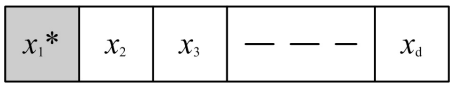
In PSO, particles will "fly" towards the gbest in the population. Both in SVOs and MVOs, PSO is used to adjust cutting planes/hyper-planes, where the cutting planes/hyper-planes move towards the best one in the population of SVO/MVO.

The detailed moving steps in SVO are described as follows:

1. In the initial phase, the concerned variable is x_1, there is one cutting plane corresponding to each particle. The particle can be shown as below.



After searching a cutting plane, the optimal x_1, which is represented by x_1^*, is found:



2. Update the personal_best $P^i_{pb}(i = 2, \ldots, d)$ of each particle and choose the one with the smallest objective value as the global_best P_{gb}. Then adjust the cutting planes according to the updating rules of PSO. The adjustment of the jth unconcerned variable of the ith particle x^i_j at time k is described in the following equations:

$$v^i_j(k+1) = wv^i_j(k) + c_1r_1(P_{gb}(k) - x^i_j(k)) + c_2r_2(P^i_{pb}(k) - x^i_j(k)), \quad (3.6)$$

$$x^i_j(k+1) = x^i_j(k) + v^i_j(k+1), \quad (3.7)$$

where $j = 2, \ldots, d$, $i = 1, 2, \ldots, M$ (M is the number of particles in the current population). w, c_1 and c_2 are all constants in standard PSO, and r_1, r_2 are random numbers in $[0, 1]$.

Since MVO only differs from SVO in the number of concerned variables, the steps of the cutting hyper-plane adjustment in MVO are similar to those of the cutting plane adjustment in SVO. In brief, move the cutting hyper-planes according to their previous performance in terms of minimal objective values found in themselves and the guide information from the best cutting hyper-plane ever achieved.

3.4.4 Operation of Integration

The motivation of integration is to retain all the useful information and combine them to create some potential solutions. This can be explained using the schema theorem and building block hypothesis [38]. A schema is a similarity template describing a subset of strings with similarities at certain string positions. It is postulated that an individual's high fitness is due to the fact that it contains good

schemata. Short and high-performance schemata are combined to form building blocks with higher performance expected. Building blocks are propagated from generation to generation, which leads to a keystone of the GA approach. Research on GA has proved that it is beneficial to preserve building blocks during the evolution process. MVOs inherit the old chromosomes from SVOs and the previous MVOs, where the building blocks likely reside. The integration of these building blocks into the initial population provides a solid foundation for the subsequent optimization.

Considering the operation of integrating MP_{k-1} with SP_k into I_k (the initial population for the kth MVO), the procedure is illustrated in Figure 3.9.

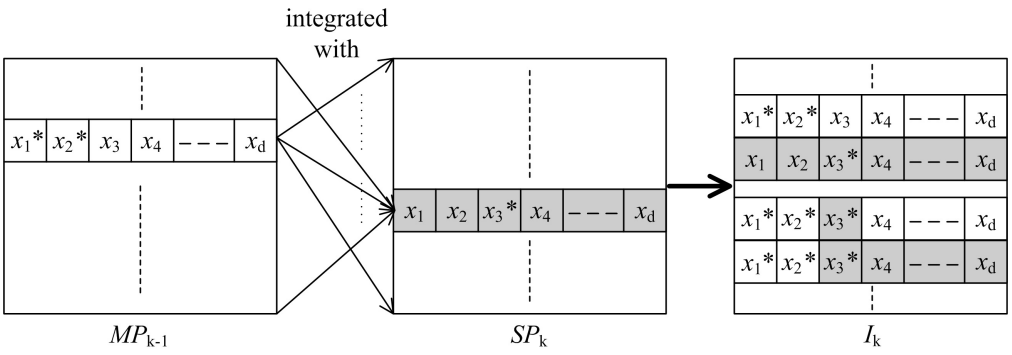


Figure 3.9: Integration operation (assume $k = 3$)

As shown in Figure 3.9, all the particles in both MP_{k-1} and SP_k are copied into I_k. Besides, for each particle in MP_{k-1}, retain its concerned variables (from x_1 to x_{k-1}), then get a value for x_k from each particle in SP_k, lastly fill up the particle from x_{k+1} to x_d with the corresponding parts of the two particles under integration. The optimal values of the concerned variables obtained earlier are marked by "*". Please note that when $k = d$, thereby the ending phase, the integration operation will be simply copying MP_{d-1} and SP_d into I_d.

3.5 Experiments

The two implementations of the proposed incremental model were tested on several benchmark problems. The experiments and their results are described in this section.

3.5.1 Performance Evaluation Metrics

For parameter optimization problems, namely, both the independent variables and the objective function are scalar quantities. The numerical values of the variables are adjusted to make the value of the objective function reach an optimum with the least computation effort [81]. Corresponding to this goal, the following metrics are used to evaluate the performance of the optimization algorithms in the present study:

1. e is the error between the optimal objective value found by an algorithm and the objective value of the true global optimum.
2. γ is the Euclidean distance between the found optimum and the true global optimum.
3. t is the computation time in second.
4. η is the rate of successful convergence: $\eta = \frac{SN}{TN}$, where SN denotes the number of trials with successful convergence ($y < 10^{-5}$) and TN denotes the total number of trials.

Besides, the standard deviations of the first three metrics are also given, which are denoted as σ_e, σ_γ and σ_t.

3.5.2 Experimental Scheme

The proposed algorithms are tested on four benchmark problems, including the Tripod function, the generalized Rastrigin function, the generalized Griewank function and the generalized Rosenbrock function. They are chosen for their characteristics that result in various difficulties for optimization algorithms. The Tripod function is discontinuous and is difficult for a lot of algorithms that are trapped in the two local minima. The generalized Rastrigin function has a large number of deep local minima that are hard to escape. The generalized Griewank function is characterized by significant interaction among its variables. The generalized Rosenbrock function is deceptively flat, thereby very difficult to locate the global optima. Given their different difficulties, it should be challenging to efficiently solve all of them with a certain algorithm based on the same parameter set. Therefore, these problems are commonly used as test problems for global optimization algorithms. Further, in order to test the scalability of the proposed algorithms, the dimensionality of the scalable test functions, including the Rastrigin, Griewank and Rosenbrock functions, are changing in the experiments. Since a problem with 30 dimensions is normally regarded a high-dimensional problem, their dimensionality varies around 30, which consists of 20, 30 and 40.

All the present results are the mean values averaged over 60 independent runs. The comparison is conducted among the PIES, the IPSO, the asynchronous version of the standard PSO (ASPSO), the PSO with charged swarms (CSPSO) and the Hybrid PSO with breeding and subpopulations (HPSO_BS). In the ASPSO, the global best is updated after each particle position updating, which differentiates it from the original standard PSO. This asynchronous version aims to improve the rate of convergence of the original one. Both the CSPSO and the HPSO_BS versions employ some concepts from other fields or algorithms to improve the standard PSO. The CSPSO introduces a repulsive force to the position updating equation,

which has an identical form to the familiar electrostatic inverse square law between charged particles. The HPSO_BS combines the reproduction and recombination of GAs, as well as the subpopulation mechanism often used to improve GAs, with the standard PSO in order to achieve a faster rate of convergence and better solutions. A t-test is completed to evaluate the significance of difference between two means, and p-value is reported for a certain claim. The significance level is set to 0.05. And $p = 0.05$ is used as the threshold for claiming significant difference, i.e., the difference is significant with at least 95% confidence level if its p-value obtained from t-test is less than 0.05.

The program of ASPSO is downloaded from one of its designers' website, `http://www.engr.iupui.edu/~eberhart/`. The CSPSO and the HPSO_BS are programmed by modifying the downloaded ASPSO program. The modifications taken for each one are set according to the descriptions presented in the corresponding papers [75–77].

For the fairness of comparison, the algorithms are given the same number of function evaluations, which is set according to experience and the configurations used in PSO context [54]. In general, this number increases as the problem gets more complicated, especially as the dimensionality increases. Normally, the parameters for heuristic optimization algorithms are set according to experiences. So the parameters in our experiments are tuned by a preprocessing procedure, whose pseudo-code is shown in Appendix A. This procedure obtains a parameter set which results in a satisfactory outcome. For the three counterpart algorithms used for comparison, the parameters are set to the values suggested in their corresponding papers. Those values have been tuned by the authors for solving the benchmark problems. The configurations are shown in Table 3.1.

With the parameter settings shown in Table 3.1, the algorithms mentioned earlier are applied to solve the chosen test problems. The experimental results are

Table 3.1: Parameter configuration

<table>
<tr><td colspan="2">Common Settings</td><td colspan="2">Swarm size: 20(Prob.1)/50(Prob.2,3,and4)</td></tr>
<tr><td rowspan="5">Specific Settings</td><td>PIES[1.]</td><td>Mutation parameters:
$c = 0.82$
$k = 5$(Prob.1)/20(Prob.2,3and4)
Iteration limitation per cutting plane:
$cp_{maxiter} = 50$</td><td rowspan="2">Dynamic inertia weight:
$w = 0.729$
Acceleration constants:
$c_1 = c_2 = 1.49445$
Number of cutting planes:
$cp = 2$</td></tr>
<tr><td>IPSO</td><td>particle number per cutting plane:
$cp_{pn} = 5$
Iteration limitation per cutting plane:
$cp_{maxiter} = 10$</td></tr>
<tr><td>ASPSO[2.]</td><td colspan="2">Dynamic inertia weight: $w = \frac{(0.9-0.4)(max\_iter-iter)}{max\_iter} + 0.4$
Acceleration constants: $c_1 = c_2 = 2$</td></tr>
<tr><td>CSPSO[3.]</td><td colspan="2">Percentage of charged particles: 50%
Electrostatic parameters: $p_{core} = 1, p = \sqrt{3}x_{max}, Q = 16$
Dynamic inertia weight: $w = 0.729$
Acceleration constants: $c_1 = c_2 = 1.494$</td></tr>
<tr><td>HPSO_BS[4.]</td><td colspan="2">Breeding rate: $pb = 0.2$
Number of subpopulations: 2
Within breeding probability: $psb = 0.6$
Dynamic inertia weight:$w = \frac{(0.9-0.4)(max\_iter-iter)}{max\_iter} + 0.4$</td></tr>
</table>

Remarks:

1. Rechenberg's 1/5 success rule for (1+1)-ES resets the mutation step size every k iterations. σ is a constant for increasing or decreasing the mutation step size.

2. *max_iter* is the iteration limitation for each run. *iter* is a counter of iteration number.

3. p_{core} and p are the lower and the upper boundary points, respectively, of the active range, in which the charged particles repulse each other. Q is the amplitude of charge of each charged particle.

4. *pb* is the probability of taking breeding for each particle. *psb* is the probability of taking breeding within the same subpopulation.

shown and analyzed in the following subsection.

3.5.3 Experimental Results and Analysis

Note that the comparison on the time cost between IPSO and PIES is given toward the end of this section, as the difference shows consistency over all those problems.

Problem 1: Tripod Function

The tripod function is a benchmark test function characterized by discontinuity, whose mathematical expression is given below [73].

$$f(x_1, x_2) = p(x_2)(1+p(x_1))+|x_1+50p(x_2)(1-2p(x_1))|+|x_2+50(1-2p(x_2))| \quad (3.8)$$

with $p(u) = \begin{cases} 1 & \text{if } u > 0 \\ 0 & \text{otherwise} \end{cases}$, $x_1, x_2 \in [-100, 100]$.

The global minimum 0 appears in (0,-50). This function is difficult for many algorithms, which tend to be trapped in the two local optima 1 and 2, sitting in (-50,50) and (50,50), respectively. The number of function evaluations is limited to 2×10^4. The results are shown in Table 3.2.

In this problem, the algorithms are often trapped in the two local minimum that are quite far away from the global minimum in the variable space. This can be seen from the results shown in Table 3.2. With the decreasing values of η, the values of γ increase rapidly and the values of σ_γ are quite large. These results indicate that the PIES and IPSO outperformed the ASPSO, CSPSO and HPSO_BS algorithms mainly in the sense that the proposed algorithms obtain the global optimum with a higher probability. The t-tests on e show the significance of difference with $p < 0.05$ for PIES vs. ASPSO and PIES vs. HPSO_BS, and for PIES vs. CSPSO. Moreover, both PIES and IPSO used less computation time than the other counterpart algorithms. The t-tests on t show the significance of

Table 3.2: Performance comparison on Tripod function

	PIES	IPSO	ASPSO	CSPSO	HPSO_BS
e	0.033384	0.083333	0.300000	1.090093	0.300021
(σ_e)	(0.181011)	(0.334039)	(0.534983)	(0.816825)	(0.534971)
γ	3.726818	7.45356	29.81424	59.77224	29.81425
(σ_γ)	(20.23868)	(28.12402)	(50.28654)	(56.61073)	(50.28653)
t	0.013921	0.013700	0.021165	1.998800	0.040186
σ_t	(0.005333)	(0.007264)	(0.009189)	(1.045945)	(0.007221)
η	95%	93.3%	73.3%	0	70%

Legends:
e: The error between the obtained optimal objective value and the objective value of the global optimum;
γ: The Euclidean distance between the found optimum and the global optimum;
t: The elapsed time of the whole optimization process measured in seconds;
η: The rate of successful convergence;
σ_e, σ_γ and σ_t are the standard deviations of e, γ and t, respectively.

the advantage for PIES over each of the three algorithms under comparison with $p < 0.01$.

Problem 2: Rastrigin Function

The Rastrigin function is widely used as a test function for optimization algorithms, and can be mathematically represented as follows [84]:

$$f(x_i|_{i=1,\ldots,d}) = 10d + \sum_{i=1}^{d}[x_i^2 - 10\cos(2\pi x_i)], x_i \in [-5.12, 5.11]. \tag{3.9}$$

The global minimum 0 is on $(0, 0, \ldots, 0)$. This scalable function has numerous deep local minima, and the number of local minima increases rapidly with the increasing dimensionality.

In order to test the searching capacity and the scalability of the proposed

algorithms in the searching spaces with different dimensionality, the algorithms were tested on the Rastrigin function with the number of variables increasing from 20, to 30, to 40. The number of function evaluations is limited to 3×10^4, 4×10^4 and 5×10^4, respectively. The results are shown in Table 3.3.

In problem 2, the algorithms hardly found the global optimum. As shown in Table 3.3, PIES and IPSO occasionally found the global optimum when the dimensionality is 20 or 30, while ASPSO, CSPSO and HPSO_BS never found the global optimum irrespective to the changing dimensionality. This is because of the huge number of deep local minima all around the landscape of the Rastrigin function, which even increases exponentially as the dimensionality increases. It can be seen from Table 3.3 that PIES and IPSO always obtain smaller values of γ and e, which indicates that the solutions found by these two algorithms are closer to the global minimum both in the variable space and in the objective space. In addition, it is observed that the advantage does not disappear with the increasing dimensionality. The t-tests on e show the significance of this advantage for PIES vs. each of the three counterpart algorithms under each dimensionality with $p < 0.01$. Besides, PIES and IPSO require less computation time than all the other three algorithms. The t-tests on t show the significance of this advantage for PIES vs. each of the three counterpart algorithms under each dimensionality with $p < 0.05$.

Problem 3: Griewank Function

The Griewank function is also a commonly used benchmark function for performance comparison among optimization algorithms, which can be mathematically described by the following equation [84].

$$f(x_i|_{i=1,\ldots,d}) = 1 + \sum_{i=1}^{d} \frac{x_i^2}{4000} - \prod_{i=1}^{d} (\cos(\frac{x_i}{\sqrt{i}})), x_i \in [-512, 511]. \tag{3.10}$$

Table 3.3: Performance comparison on Rastrigin function

	d	PIES	IPSO	ASPSO	CSPSO	HPSO_BS
e (σ_e)	20	10.63978 (3.301476)	11.63824 (2.493062)	23.17084 (6.812518)	135.3073 (16.12061)	17.86163 (6.253198)
	30	24.874 (2.112468)	27.85887 (2.365542)	46.98929 (14.54243)	221.3652 (26.21792)	44.14434 (13.56359)
	40	48.83226 (2.781271)	52.73416 (2.835612)	83.25465 (25.10655)	220.2099 (18.70921)	82.30526 (22.41407)
γ (σ_γ)	20	3.155609 (0.635421)	3.320192 (0.699123)	4.740553 (0.718565)	5.991533 (0.951820)	4.141190 (0.728472)
	30	4.974823 (0.782611)	5.264857 (0.790011)	6.730022 (1.044851)	7.308517 (0.965716)	6.47033 (0.968955)
	40	6.879697 (0.743824)	7.243243 (0.792230)	8.872394 (1.309052)	8.202101 (1.336749)	8.715342 (1.063300)
t (σ_t)	20	0.235200 (0.018253)	0.188033 (0.016914)	0.280973 (0.026622)	10.19891 (2.902710)	0.391906 (0.027932)
	30	0.353333 (0.011982)	0.296833 (0.018036)	0.371000 (0.019184)	35.18425 (5.118960)	0.631000 (0.012801)
	40	0.522233 (0.013926)	0.480667 (0.014482)	0.451000 (0.018953)	56.19886 (5.980733)	0.715333 (0.015329)
η	20	6.7%	3.3%	0	0	0
	30	3.3%	3.3%	0	0	0
	40	0	0	0	0	0

Legends: Refer to Table 3.2.

The global minimum 0 is on $(0, 0, \ldots, 0)$. The difficulty of this scalable function is the nonlinear interactions among its variables. This means that in the SVOs related to those correlated variables, searching in cutting planes or adjusting the positions of these cutting planes may result in performance improvement but drive solutions away from the global optimum.

The algorithms were tested on the function with the number of variable increasing from 20, to 30, to 40. The number of function evaluations is limited to 3×10^4, 4×10^4 and 5×10^4, respectively. The results are shown in Table 3.4.

In problem 3, PIES and IPSO mostly obtained a successful convergence rate around 50% no mater what the dimensionality is. This may be because of a flaw of Griewank function when scaled. Its summation term induces a parabolic shape while the cosine function in the product term creates "waves" over the parabolic surface creating local minima. But as the dimensionality of the search space is increased, the contribution of the product term is reduced and the local minima become quite small. It is easy to escape from the small minima although the number of them increases. In other words, increasing the dimensionality not necessarily increases the difficulty. Even so, the rate of successful convergence of ASPSO reduces from 10% to 0 when the dimensionality increases from 20 to 30 and 40. CSPSO is even worse, whose η values are always 0. There is an interesting finding that the performance of HPSO_BS improves when the dimensionality increases from 20 to 30 and 40. This may indicate that the breeding mechanism employed by HPSO_BS gets more effective when the local minima become smaller. However, the performance of PIES and IPSO still outperforms HPSO_BS in terms of all the metrics. Besides, PIES and IPSO used less computation time than all the other three algorithms. The t-tests on both e and t show the significance of these advantages for PIES vs. each of the three counterpart algorithms under each dimensionality with $p < 0.01$.

Table 3.4: Performance comparison on Griewank function

	d	PIES	IPSO	ASPSO	CSPSO	HPSO_BS
e (σ_e)	20	0.004034 (0.005692)	0.0055733 (0.004649)	0.033078 (0.026430)	1.023099 (0.023316)	0.027861 (0.019253)
	30	0.005942 (0.005112)	0.006339 (0.005106)	0.043452 (0.025532)	1.067502 (0.008917)	0.010571 (0.013392)
	40	0.005991 (0.005813)	0.006300 (0.006159)	0.687660 (0.212764)	1.113059 (0.018437)	0.011125 (0.017212)
γ (σ_γ)	20	2.416988 (3.284386)	2.761449 (3.277114)	10.24552 (5.027050)	12.45743 (2.964655)	9.570022 (4.481634)
	30	2.727543 (3.086577)	3.423879 (3.148333)	5.074749 (4.433809)	16.40414 (1.070288)	4.845311 (4.392626)
	40	2.750714 (2.933011)	3.770498 (3.169240)	8.180290 (2.397253)	21.19193 (1.765187)	4.507249 (4.970638)
t (σ_t)	20	0.182167 (0.009920)	0.171218 (0.008806)	0.201884 (0.011449)	8.469578 (1.46494)	0.298828 (0.018878)
	30	0.312141 (0.010120)	0.297853 (0.011630)	0.344031 (0.012660)	12.35906 (1.24593)	0.422328 (0.014220)
	40	0.516654 (0.012420)	0.484014 (0.018830)	0.599104 (0.013440)	17.84344 (1.15077)	0.703672 (0.012844)
η	20	61.7%	53.3%	10%	0	15%
	30	53.3%	50%	0	0	36.7%
	40	53.3%	51.7%	0	0	40%

Legends: Refer to Table 3.2.

Problem 4: Rosenbrock Function

The Rosenbrock function is a famous benchmark test function for the deceptive flatness of its landscape. Its mathematical expression is shown below.

$$f(x_i|_{i=1,\ldots,d}) = \sum_{i=1}^{d-1}[(1-x_i)^2 + 100(x_i^2 - x_{i+1})^2], x_i \in [-10, 10]. \tag{3.11}$$

For this function, the global minimum 0 is on $(1, 1, \ldots, 1)$. This scalable function is deceptively flat and there are nonlinear interactions among its variables. The algorithms were tested on this function with the number of variables increasing from 20, to 30, to 40. The number of function evaluations is limited to 3×10^4, 4×10^4 and 5×10^4, respectively. The results are shown in Table 3.5.

In this problem, the algorithms never found the global minima, resulting from the deceptively flat landscape of the Rosenbrock function. It can be seen from the results shown in Table 3.5 that the e values of the solutions with similar γ values may have big gaps, and smaller γ not necessarily corresponds to smaller e. These phenomena are decided by the characteristics of this problem, which can be illustrated by the following 2D example. Assume two solutions $\mathbf{x}_1(2, 1)$ and $\mathbf{x}_1(2, 4)$. Their Euclidean distances to the global minimum (1,1) are $\gamma_1 = 1$ and $\gamma_2 = \sqrt{10} \approx 3.16$, while their errors in the objective space are $e_1 = 901$ and $e_2 = 1$, respectively. In view of this, it can be understood why PIES and IPSO have obvious superiority over the counterpart algorithms in the objective space, but have similar (or slight better) values of γ with (than) them. The t-tests on e show the significance of this advantage for PIES vs. each of the three counterpart algorithms under each dimensionality with $p < 0.01$. What should be noted is that PIES and IPSO used more computation time than ASPSO and HPSO_BS. When the dimensionality is 20, the t-tests on t show no significant difference for PIES/IPSO vs. ASPSO or HPSO_BS. When the dimensionality is either 30 or 40, the t-tests on show significant difference for PIES vs. ASPSO with $p < 0.01$, but

Table 3.5: Performance comparison on Rosenbrock function

	d	PIES	IPSO	ASPSO	CSPSO	HPSO_BS
e (σ_e)	20	13.30043 (0.815059)	17.53057 (0.878382)	33.22295 (1.164540)	788.9359 (496.1137)	18.14104 (0.888727)
	30	22.54812 (0.929052)	26.01717 (1.088785)	44.04917 (11.56554)	1489.816 (861.9554)	38.54502 (6.08283)
	40	32.17153 (0.977143)	36.64323 (1.124466)	84.83606 (18.28156)	2354.795 (1308.792)	60.37172 (6.320782)
γ (σ_γ)	20	3.731704 (0.085737)	3.974769 (0.103466)	4.142770 (0.135447)	4.564797 (0.668355)	3.997174 (0.106187)
	30	4.830876 (0.088581)	5.081101 (0.095936)	5.299149 (0.278508)	5.431379 (0.763239)	5.169788 (0.197905)
	40	5.730371 (0.092667)	6.083316 (0.093163)	6.075301 (0.295781)	6.225448 (0.814495)	6.176902 (0.188174)
t (σ_t)	20	0.194554 (0.015402)	0.189823 (0.010792)	0.172891 (0.012696)	17.85972 (1.428040)	0.188469 (0.013187)
	30	0.316233 (0.010644)	0.291533 (0.012713)	0.209348 (0.011827)	23.03181 (1.187610)	0.282156 (0.016349)
	40	0.462498 (0.011859)	0.426867 (0.013258)	0.328234 (0.017790)	30.891621 (1.354241)	0.399220 (0.015249)
η	20	0	0	0	0	0
	30	0	0	0	0	0
	40	0	0	0	0	0

Legends: Refer to Table 3.2.

no significant difference for PIES vs. HPSO_BS. An interesting observation is that ASPSO and HPSO_BS required less time than IPSO/PIES. This exceptional result may be due to their convergence at a very early iteration. After this premature convergence, the operation of updating pbests and gbest stops so that the time cost could be dramatically reduced.

In all the analysis above, PIES and IPSO are considered as a whole because they have similar performance. To further differentiate between their performance, their performance on the test problems are compared. It is found that PIES slightly outperformed IPSO in terms of e and γ on all the four test problems, while in most cases IPSO needs less computation time than PIES. However, the t-tests on both e and γ show no significant difference for PIES vs. IPSO in the first three problems. Only in problem 4, the t-tests on e show the significance of the superiority of PIES over IPSO under each dimensionality case with $p < 0.01$. This significant superiority may be due to the mutation of the (1+1)-ES used in PIES. This mutation results in movements inside hyper-spheres with adaptive size. The size of the hyper-sphere increases with the mutation failure increases, which may help avoid premature convergence on the deceptively flat landscape. This finding suggests using PIES to handle optimization problems with flat landscapes other than IPSO.

In summary, the results show that:

1. Compared to the three counterpart algorithms, the two incremental algorithms obtained larger rates of successful convergence as well as better solutions closer to the global optimum, with a shorter computation time.

2. Although PIES outperformed IPSO with more computation time in most cases, there is no statistical significant difference between their performances except in Problem 4.

In next section, these findings are discussed.

3.6 Merits of Incremental Global Optimization

The experimental results suggest that the proposed incremental algorithms are superior to their counterparts on the benchmark functions tested. In all the test cases, the incremental algorithms improved the performance of the standard PSO, and obtained better or comparable results compared to other modified PSO algorithms. Their merits that may result in the satisfactory outcome are discussed in this section.

1. Contribution of SVO

   Two assumptions are made as follows:

   (a) In total, d SVOs are conducted for a d-dimensional problem.

   (b) The probabilities of successfully hitting the global optima in the SVOs, named *success probability*, are $p_1, p_2, \ldots, p_d$, respectively.

   Note that according to the definition of success probability, success in a SVO not only requires the values of concerned variables are optimal but also the unconcerned variables.

   Let S denote the number of successful SVOs in which global optimum is found. Since success in any SVO leads to the success to find the global optimum, the probability of $S \geq 1$ is the success probability with d SVOs, denoted by p_{SVOs}. This means in equivalence that failure to find the global optimum results from the joint failure of SVOs, which is described by Equation (3.12).

$$p_{SVOs} = P(S \geq 1) = 1 - P(S = 0) = 1 - \prod_{i=1}^{d}(1 - p_i). \tag{3.12}$$

Since usually $p_i \ll 1$, Equation (3.13) can be derived from Equation (3.12).

$$p_{SVOs} = 1 - \prod_{i=1}^{d}(1 - p_i) \approx 1 - (1 - \sum_{i=1}^{d} p_i) = \sum_{i=1}^{d} p_i. \tag{3.13}$$

According to Equation (3.13), it can be concluded that the success probability with all the SVOs is approximately equal to the sum of the success probability of each SVO. This conclusion may suggest the use of multiple SVOs.

From another point of view, the use of SVOs would help maintain population diversity, which are initialized and operated independently. The SVOs' results are integrated into the main population which goes through MVOs in every phase. In other words, there are new particles continuously joining the population from the beginning to the end of the incremental optimization process. This could avoid premature convergence to some extent. The stated ability of maintaining population diversity can be seen from the following example. Figure 3.10 shows the best solution (gbest) of every iteration during one run of IPSO on problem 1.

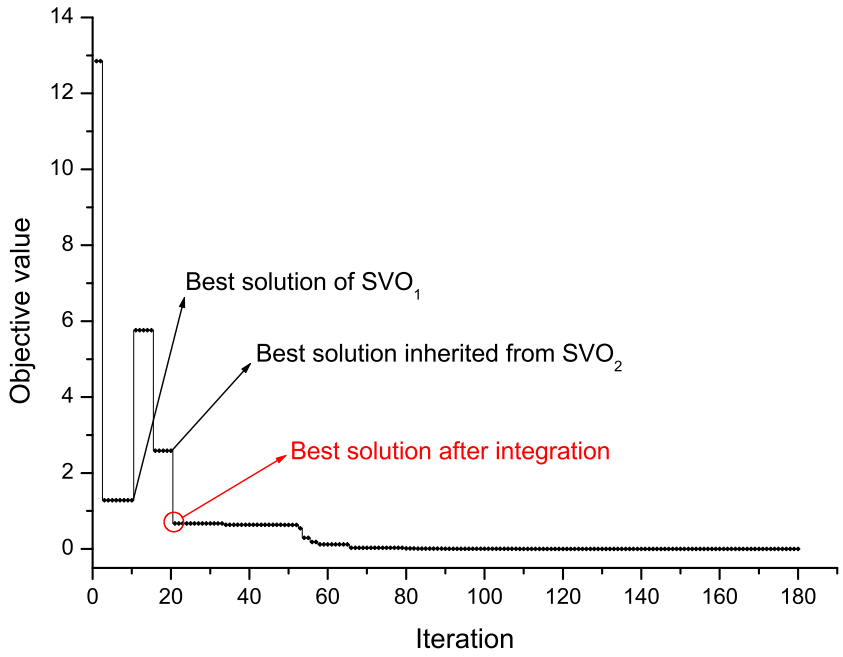


Figure 3.10: Trace of the gbest during one run of IPSO

Figure 3.10 Trace of the gbest during one run of IPSO As shown in Figure 3.10, SVO1 is trapped in one of the two local minima of the Tripod function, which is 1. Although SVO2 is trapped in the other local minima 2, it maintains the diversity of the population. This diversity maintenance helps IPSO escape from the local minima through integration, which will be discussed next.

2. Effect of integration

   The integration procedure collects information from the SVO in the current phase and the MVO in the last phase, which is used to produce more promising individuals. These individuals act as initial solutions for the current MVO. In this way, useful information is inherited and propagated. Considering the example described above in the first point, although the SVO_1 and the SVO_2 are trapped in the two local minima, the integration taken on the solutions inherited from them provides a more promising solution. This solution helps IPSO escape from the local minima, which acts as an initial solution in MVO_1.

   The good effect of integration may be explained using the schema theorem and building block hypothesis [38]. A schema is a similarity template describing a subset of strings with similarities at certain string positions. It is postulated that a chromosome's high fitness is due to the fact that it contains good schemata. Short and high-performance schemata are combined to form building blocks with higher performance expected. Building blocks are propagated from generation to generation, which leads to a keystone of the GA approach. Research on GA has proved that it is beneficial to preserve building blocks during the evolution process. A particle is like a chromosome and its variables are like the genes of a chromosome. By analogy, the combinations of variable values resulting in good fitness can be seen as building block accumulation. That is to say, the integration inherits building blocks from

SVO and MVO and propagates them, which would improve the convergence of the incremental algorithms.

3. Necessity of MVO

   MVO differs from SVO mainly in two aspects. One is that part of the initial population is provided by the integration. So MVOs could focus their search in some promising area. The other one is that more than one variable is involved in the fine tuning, namely local search. This characteristic is especially beneficial when the variables are coupled, namely non-separated. Without MVOs, the coupled variables have no chance to be fine tuned together.

Additionally, the updating operations corresponding to one function evaluation for cutting plane/hyper-plane searching in a SVO and a MVO of intermediate phase only involve part of the variables, as the operations are only taken on the concerned variables of the SVO/MVOs which do not contain all the variables. In contrast, one function evaluation in the non-incremental algorithms corresponds to those updating operations over the whole variable set. This difference may be the reason why given the same number of function evaluation, the incremental algorithms generally cost less computation time.

What should be noted is that the discussion above is based on the incremental model, no specific algorithm is involved. In fact, this study aimed at proposing an incremental procedure for optimization. Generally, this incremental procedure would be able to apply to any algorithm and result in different incremental algorithms. In the present study, PSO was chosen as the vehicle to implement the incremental procedure, resulting in IPSO. Furthermore, to show various possibilities of implementing the incremental procedure, (1+1)-ES was employed to replace PSO in the component of fine tuning, resulting in PIES. The comparison results show that this replacement do make differences, although the t-tests show no statistical significance of the differences in most cases. In future, we will further study

various implementations of the proposed incremental model. Different combination of algorithms will be investigated to reveal its benefits.

Some readers may argue that according to the "No Free Lunch" (NFL) theorem all algorithms that search for an extremum of a cost function perform exactly the same, when averaged over all possible cost functions [82]. However, the NFL emphasizes the set of all functions. It does not necessarily hold for all subsets of this set. For example, Christensen et al. constructed a set of "searchable" functions, on which a proposed general algorithm can be proved performing better than random search [83]. In this present study, the empirical results may indicate that the incremental algorithms perform, on average, better than others over a limited subset of the set of all functions. No further attempt will be made to characterize this subset in this paper, which may be done in future. Instead, it is considered that on what function the proposed incremental algorithms could not work well. An implicit assumption of the cutting plane mechanism, based on which the incremental model is proposed, is that the quality of a solution will more or less reflect the quality of its neighbor area. That is why cutting planes/hyper-planes are adjusted according to the quality of the solutions found on them. However, this assumption does not hold for functions with singular points or discontinuity. This implies that the incremental algorithms may be not suitable for those functions.

The merits of the proposed incremental model have been concretely exhibited by applying it to PSO. To further investigate the possibility of improving this model, a parallel implementation of PSO is devised and studied in the next chapter.

Chapter 4

Parallel Incremental Particle Swarm Optimizer

In the last chapter, novel incremental global optimization algorithms, the IPSO and PIES, have been developed and discussed. In order to enhance the profitability of information sharing among SVOs used in them, a parallel implementation is studied in this chapter, which performs searching in different spaces with reduced dimensionality and implement information sharing among the searching mechanisms. The basic procedure of the parallel implementation is the same no matter it is for IPSO or PIES. Without the loss of generality, only the parallel implementation of IPSO is investigated for the ease of description, as the number of component algorithms used in IPSO is less than in PIES. The resulting parallel IPSO (PIPSO) makes progress in two aspects compared with the original IPSO. Principally, it enhances information sharing by employing a Bulletin Board System (BBS), which collects information from individual SVOs to broadcast to all the other SVOs. As an accessory effect, PIPSO makes it easy to distribute the incremental optimization process to a parallel computing system. Performance comparison demonstrates the efficacy of the parallel model.

4.1 Motivation

As stated in Chapter 3, the incremental model is applied to PSO to overcome its flaw, resulting in IPSO. The basic idea of information sharing adopted in incremental algorithms is to distribute the subcomponents of individual solutions over a set of subpopulations. For example, a function optimization problem with d variables is partitioned into d subpopulations, each subpopulation corresponding to one of the variables. Potential solutions to the optimization problem are constructed by taking representatives from each of the d subpopulations to build a d-dimensional vector. The different subpopulations are thus cooperating to find the solution, since no single subpopulation has the necessary information to solve the problem by itself. However, a deficiency of IPSO may be that its information flow proceeds in a serial manner, which means information regarding different dimensions cannot be shared directly or mutually. More specifically, the concerned variable of good solutions found in a SVO may indicate promising positions along the corresponding dimension. If this information can be used to generate cutting planes for other SVOs, promising search spaces could be explored earlier and more thorough. However, good solutions found in a certain SVO of IPSO do not have an opportunity to communicate with other SVOs, instead, the best of them is sent to main information flow when the flow comes to this SVO. Therefore, employing a more effective information sharing mechanism would further improve IPSO.

4.2 Implementation of PIPSO

As stated above, a parallel version of IPSO, PIPSO, is proposed in order to improve the efficiency of IPSO. The detailed implementation of the PIPSO is described in this section.

4.2.1 Procedure of PIPSO

The PIPSO is designed to optimize particles along individual dimensions and search the whole input space through information sharing. The basic ideas of PIPSO can be described as follows:

1. To explore each dimension of the problem independently in SVOs, the concerned variable of particles is evolved while the unconcerned ones are fixed. That is to say, several cutting planes are searched simultaneously.

2. To select the best particle from each SVO after every run of cutting plane search. The fitness value of this particle is decided by evaluating the geographic relationship between this particle and other particles in the same SVO.

3. To post the selected individual with its fitness to an information board, which is called BBS.

4. To generate cutting planes for next epoch for all the SVOs by using the information posted on the BBS.

Based on these ideas, the procedure of the PIPSO can be shown in Figure 4.1.

As shown in Figure 4.1, particles in SVOs are still in charge of searching along their individual dimensions like in IPSO. The information sharing is denoted by the lines between SVOs and the BBS. The line starting from a SVO stands for posting the best solution found in this SVO to the BBS, and the lines starting from the BBS stands for broadcasting the collected information to the other SVOs.

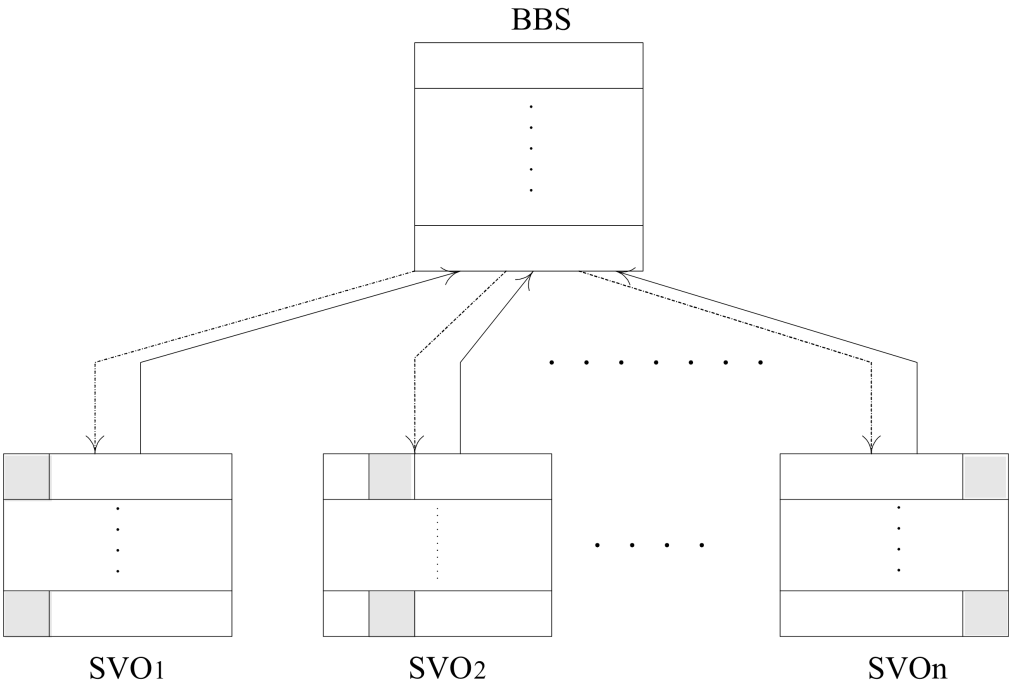


Figure 4.1: Procedure of PIPSO

4.2.2 Fitness Assignment Methods

Particles posted on the BBS are used to generate cutting planes for SVOs, which are expected to locate in some promising area. The fitness values of these particles should reflect their capability of generating cutting planes in promising areas. From an intuitional point of view, objective values of the posted solutions reflect their qualities, which can be used to evaluate their fitness. Additionally, the geometric relationship regarding the concerned variable among the solutions found by searching every cutting plane in a SVO is also a crucial factor for deciding the fitness of the particle posted by this SVO. If the concerned variables of all the solutions in the SVO are close, it could be inferred that the cutting planes converge to some extent. In this case, it may imply that the best solution of this SVO is in a promising area that needs more exploration. Thus, to design a method for fitness assignment, we should take into account these two factors, the objective values of the individuals on the BBS and the distribution of the values of the concerned variables of the solutions found in a SVO.

According to the discussion above, the evaluation equations of the two factors contribute to the fitness value of an individual on the BBS. The one involving the objective value is denoted by fac_{obj}, while the one involving the concerned variable is denoted by fac_{var}. The evaluation equations are proposed accordingly as follows:

$$fac_{obj} = \frac{r}{b(b-1)} + \frac{0.5}{b}, \tag{4.1}$$

$$fac_{var} = \frac{\frac{1}{|\Omega|}\sum_{p\in\Omega}|x_k^{p,*} - x_k^{*,*}|}{VarRg_k}, \tag{4.2}$$

where $r \in [0, d-1]$ is the rank of an individual on the BBS decided by its objective value (smaller objective value corresponds to lower rank), b is the number of individuals on the BBS which is equal to the number of variables, Ω stands for the collection of solutions found from all the cutting planes of a SVO and $|\Omega|$ is the size of Ω which should be one less than the number of cutting planes in a SVO. In Equation (4.2), $x_k^{p,*}$ is the concerned variable of the solution found from the pth cutting plane in the kth SVO, $x_k^{*,*}$ is the concerned variable of the best solution in the kth SVO that will be posted to the BBS and $VarRg_k$ is the range of the concerned variable of the kth SVO. As a convention in evolutionary computation, the fitness value is proportional to the quality of an individual and usually normalized to 1. Following this convention, two components are defined to evaluate the fitness of individuals on the BBS based on the two factors, respectively:

$$fit_1 = 1 - fac_{obj} = 1 - \Big[\frac{r}{b(b-1)} + \frac{0.5}{b}\Big], \tag{4.3}$$

and

$$fit_2 = 1 - fac_{var} = 1 - \frac{\frac{1}{|\Omega|}\sum_{p\in\Omega}|x_k^{p,*} - x_k^{*,*}|}{VarRg_k}. \tag{4.4}$$

Using fit_1 is more or less like a gradient descent technology, which promotes exploration and may cause premature convergence. In contrast, employing fit_2 tends to

give more credit on dimensions with less fluctuation, along which a search may be less likely to be trapped at local optimum. Based on these two factors, two fitness assignment methods are designed to balance exploration and exploitation.

1. Combined Balance (CB): $fit = \frac{fit_1+fit_2}{2}$ is used throughout the whole procedure. In this case, fit_1 and fit_2 play the same role in deciding the fitness value.

2. Sequential Balance (SB): The whole searching procedure is split into two equal parts. fit_1 is used for the first half and fit_2 is used for the second half. This method is following a well-known principle that exploration is preferred at the beginning of searching for fast convergence, while exploitation needs to be emphasized at a later stage in order to prevent premature convergence.

The effect of the fitness assignment method on the performance of PIPSO will be investigated by experiments.

4.2.3 Roulette Wheel Selection on BBS

Besides the fitness assignment, how to use the information on the BBS to generate cutting planes for SVOs is another important issue. Each individual on the BBS is the best solution from the corresponding SVO, which is eligible to generate cutting planes for other SVOs. A selection operation is performed to decide which individual to be used. The task of the selection is to make SVOs search on promising cutting planes and avoid being trapped in any cutting plane. This scenario is of similar function with the selection performed in GAs, which chooses chromosomes with high fitness to propagate without loosing diversity. So the selection schemes used by GAs are used on the BBS.

Roulette wheel selection is the simplest selection scheme used in the field of GAs, which is a stochastic process analogous to a roulette wheel with each slice proportional in size to the fitness. When performing the roulette wheel selection, the individuals are mapped to contiguous segments of a line, such that each individual's segment is equal in size to its fitness. Subsequently, a random number is generated and the individual whose segment spans the random number is selected. The process is repeated until the desired number of individuals is obtained. With roulette wheel selection, the fittest member is not guaranteed to be selected, merely has a very good chance. Such feature matches the requirement of selecting individuals with high fitness from the BBS to generate cutting planes without being trapped in any single cutting plane. Thus, the roulette wheel selection is applied to the individuals on the BBS and the selected ones are used to generate cutting planes for the SVOs in the next iteration.

What should be noted is the relationship between the number of cutting planes in each SVO, denoted by NC, and the number of individuals on the BBS, denoted by NB. For SVO_i, all the individuals on the BBS except the one posted by itself are candidates to generate its cutting planes, as the posted cutting plane has already been searched. Since each SVO posts its best solution to the BBS, SB equals to the number of SVOs, namely the dimensionality of the problem d, which means there are totally $d-1$ candidates to generate cutting planes for any SVO. The generation of cutting planes corresponding to various relationships between the number of candidates and NC is shown in Figure 4.2. If the number of candidates is greater than the number of cutting planes, the roulette wheel selection described above is conducted. If they are equal, all the candidates are used to generate cutting planes. Otherwise, besides using all the candidates on the BBS, the rest of the cutting planes are generated randomly.

Since the incremental optimization algorithms are developed to solve difficult problems where high dimensionality is usually the major difficulty, d is probably a

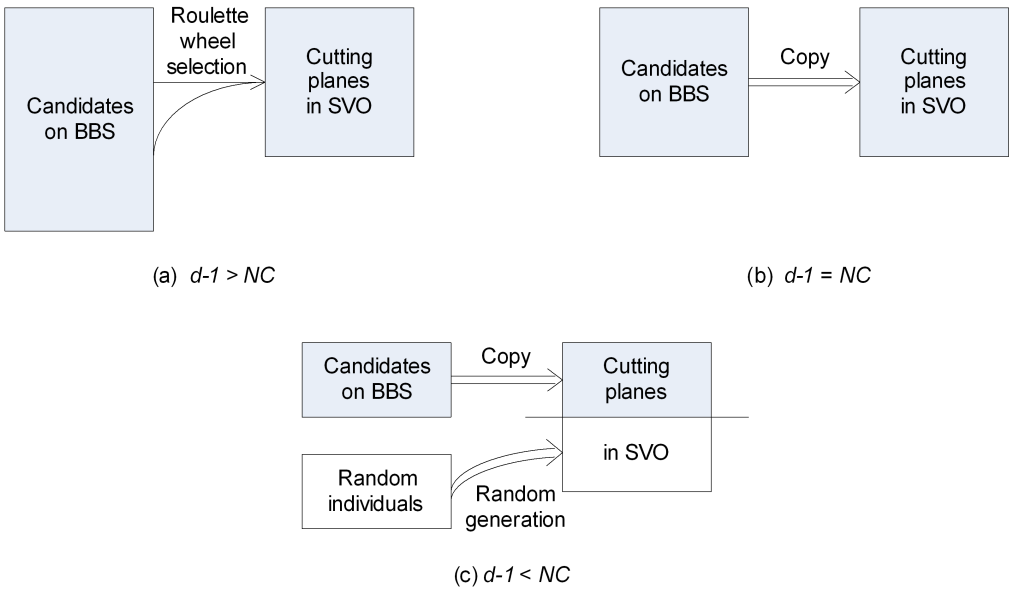


Figure 4.2: Generation of cutting planes for SVOs

large number. In this case, the number of cutting planes used in each SVO should be small in order to constrain computational cost. Thus, $d-1$ is usually greater than NC, which is the situation shown in Figure 4.2 (a).

4.2.4 Mutation Operator

The SVOs, in case of posting the same individual, would be trapped because the candidates for roulette wheel selection are the same. To avoid this kind of premature convergence, a mutation operator is adopted, which takes place when the variance of the objective values of the individuals on the BBS is less than some predetermined threshold. The rationale behind this mechanism is described as follows. If the variance is very small, it is likey that all the individuals on the BBS has converged to one solution. Without mutation, the cutting planes of each SVO assigned for the next iteration would be the same, so that the SVOs hardly achieve any progress in the following search. Therefore, when the variance decreases to certain extent, mutation is necessary to keep the diversity of the BBS. The pseudo-code of the mutation operator is show in Figure 4.3.

FOR (each particle in every SVO)

IF $rand < \left(1 - \frac{curiter}{totiter}\right)^{1/mutrate}$

{

wichdim = random(unconcerned variable)

mutrang =

$$\left(upperbound\left[wich\,\dim\right] - lowerbound[wich\,\dim]\right) * \left(1 - \frac{curiter}{totiter}\right)^{1/mutrate}$$

ub = particle[wichdim] + mutrang

lb = particle[wichdim] − mutrang

particle[wichdim] = realrandom (lb, ub)

}

Legends: 1) *curiter* is the current iteration number and *totiter* is the total number of iteration;
2) *rand* is a random number and *mutrate* is the mutation rate between 0 and 1;
3) *wichdim* is a randomly selected unconcerned variable;
4) *mutrang* is the mutation range of wichdim;
5) *ub* and *lb* are upper and lower bounds of wichdim after mutation, respectively;
6) *particle* is the particle to be mutated.

Figure 4.3: Pseudo-code of the mutation operator

The value of *mutrate* needs to be selected to achieve the goal that all the particles in the population are affected by the mutation operator (as well as the full range of the decision variables) at the beginning. As the number of iterations increases, the effect of the mutation operator decreases, which aims to guarantee convergence at the end. That is to say, as the number of iteration increases both the number of affected cutting plane and the mutation range will decrease rapidly by using a nonlinear function.

4.3 Comparing PIPSO with IPSO and CPSO

Since PIPSO is developed from IPSO, they certainly have similar elements. Besides, PIPSO shares some common features with the CPSO [54] with respect to its parallel structure. The common and distinct aspects among these three PSO-based algorithms are compared in Table 4.1. As a parallel version of IPSO, PIPSO

Table 4.1: Feature comparison among PIPSO, IPSO and CPSO

Features	PIPSO	IPSO	CPSO
SVO	Employing multiple cutting planes	Employing multiple cutting planes	Employing single cutting plane
MVO	×	✓	×
Information sharing among SVOs	Information collection and broadcasting by a BBS	Integration between SVO and MVO from adjacent phases	Information transfer through a context vector
Mutation	✓	×	×

keeps the basic element, SVO. But PIPSO and IPSO employ different information sharing mechanism. PIPSO collects best solutions from SVOs and utilizes them to generate cutting planes for SVOs in the next iteration, while IPSO integrates solutions from SVO in current phase and MVO in previous phase to generate cutting hyper-planes for the MVO in current phase. Given a d-dimensional problem, IPSO should conduct $d-1$ MVOs that result in high computational cost, especially when d is a large number. Besides, SVOs in IPSO are mutually isolated without information exchange so that the optimized values of the concerned variable can not be used to generate cutting planes, which is inefficient in making use of obtained information. By using BBS, PIPSO overcomes the two drawbacks of IPSO stated above. However, taking MVOs off results in losing chances to adjust more than one variables simultaneously, which may cause being trapped in local optima. That is the reason why mutation is applied in PIPSO, which help escape from local optima. An example is given in Figure 4.4.

In the above figure, B stands for the only solution in BBS, G stands for the true global best and the crosses stand for the particles in SVOs. It can be seen that the particles will finally converge to B. They could not get to G as they can only move along the two axes. Mutation is a possible way to jump out.

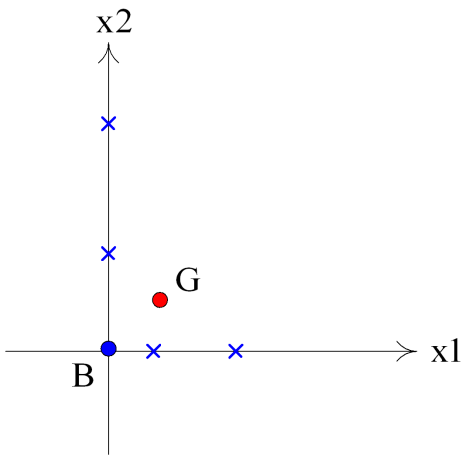
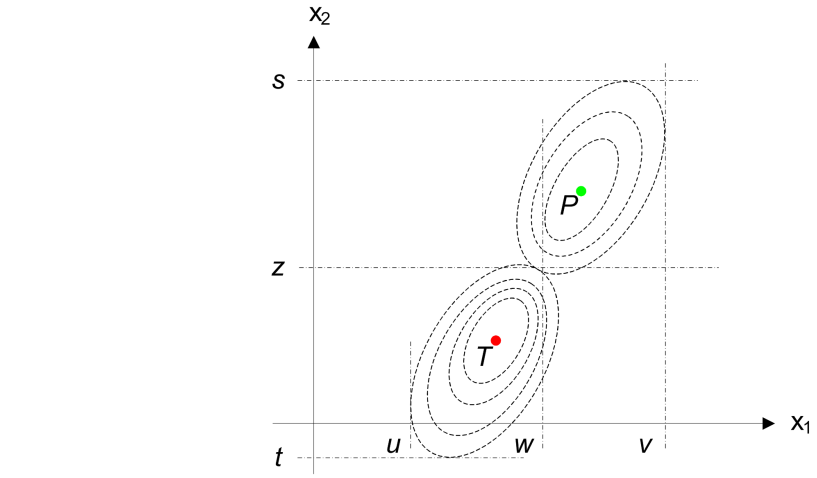


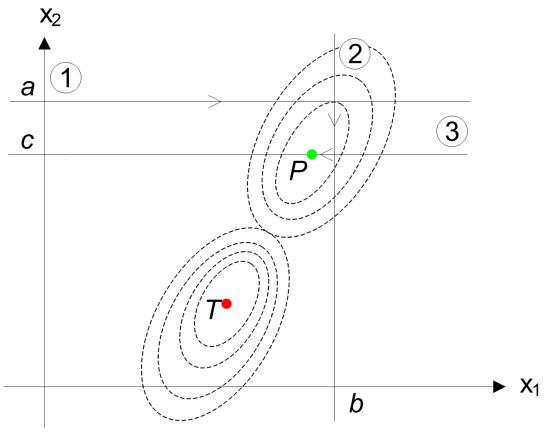
Figure 4.4: Illustration of parallel PSOs drawback

The construction of CPSO seems similar to that of PIPSO. In fact, they are essentially different in the sense that SVOs in CPSO proceed in a serial manner with information transferred from one SVO to another, while SVOs in PIPSO perform searching in a parallel manner and share information altogether. This difference and its consequence can be illustrated by a 2D problem in Figure 4.5. Here, the ellipses stand for contour lines of the landscape of the objective function and numbers insides the circles is the sequence number of iteration, u and v are the upper and lower bounds of x_1, and s and t are those of x_2. It is assumed that all the points on a contour line have equal objective values, the objective value decreases as the ellipse shrinks and the difference between two adjacent ellipses is a fixed value. In addition, P and T are two local optima. The gradient of the ellipses converge to T is greater than to P, which means T is the global optimum. As shown in Figure 4.5 (i), it is further assumed that the line $x_1 = w$ is the common tangent line of two ellipses with the same objective value, surrounding P and T, respectively, and $x_2 = z$ is also such a line.

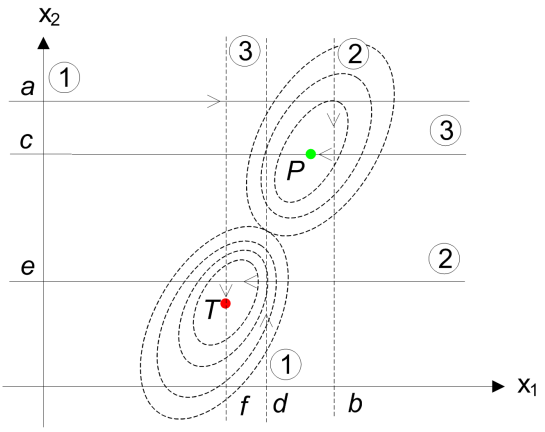
Assume there is only one particle in the swarm for both CPSO and PIPSO (see Figure 4.5 (ii)). CPSO starts from searching along x_1 with x_2 being fixed to a, which is exactly what is called cutting plane in this thesis. Suppose the found optimal value of x_1 is b. Then the value b will be filled into the corresponding



(i) Landscape of a 2D example



(ii) CPSO



(iii) PIPSO

Figure 4.5: Illustration of CPSO and PIPSO

position in the "context vector", which guides CPSO to search along x_2 for the optimum. Please note that although CPSO may need more than three steps as what is shown in the graphical illustration, the convergence to P is inevitable if the 1D search can always successfully find true optimum, as the trajectory will proceed either horizontally or vertically with greatest gradient descent in that case. In other words, the probability of obtaining T is in equivalence to the probability of intersecting between z and t by the first randomly generated cutting plane, which is

$$P_{CPSO} = \frac{z-t}{s-t}. \tag{4.5}$$

For comparison, the operation of PIPSO is illustrated in Figure 4.5 (iii), where solid lines stand for cutting planes in SVO_1 and dashed straight lines stand for cutting planes in SVO_2. It can be seen that SVO_1 and SVO_2 operate simultaneously and exchange their obtained information. As analyzed above, T would be achieved as long as a cutting plane in SVO_1 intersects between z and t or a cutting plane in SVO_2 intersects between u and w. That is to say, the probability of obtaining T shall be

$$P_{PIPSO} = \frac{z-t}{s-t} + \frac{w-u}{v-u}. \tag{4.6}$$

From the above analysis, it can be concluded that the probability of successfully finding the global optimum by PIPSO is larger than by CPSO with the same swarm size. The rationale behind this predominance is that generating cutting planes along different dimensions would result in different types of difficulties for approaching the region surrounding global optimum, as the complexity of the orthographically projected landscape varies with projecting direction. PIPSO collects the success probabilities of all the SVOs as it proceeds from all the dimensions in a parallel manner, while the starting dimension decides the success probability of CPSO as it serially searches the dimensions in a fixed order.

4.4 Experiments

In order to find out the performance of the PIPSO, it was tested on several benchmarked problems. The experiments and their results are described in this section.

4.4.1 Performance Evaluation Metrics

Since PIPSO is designed to improve IPSO, the metrics that were used to evaluate the performance of IPSO in Chapter 3, are still used in this chapter for consistence:

1. e is the error between the optimal objective value found by an algorithm and the objective value of the true global optimum.

2. γ is the Euclidean distance between the found optimum and the true global optimum.

3. t is the computation time in second.

4. *eta* is the rate of successful convergence: $\eta = \frac{SN}{TN}$, where SN denotes the number of trials with successful convergence ($y < 10^{-5}$) and TN denotes the total number of trials.

The standard deviations of the first three metrics are given as σ_e, σ_γ and σ_t.

4.4.2 Experimental Scheme

The experiments performed in this chapter aim at investigating several key properties of PIPSO, including its convergence property and robustness. A comparison among PIPSO, IPSO, CPSO and standard PSO investigates the efficacy of these algorithms on several benchmark functions. A validation of some key parameters of the PIPSO studies the robustness of the scheme.

Although limited experiments may not give accurate indication of the performance of an algorithm, they can reveal certain aspects of the algorithm considering the characteristics of the test functions. PIPSO may be more sensitive to the correlations among variables than IPSO, as the MVOs in which different variables are searched together are replaced by the BBS when modifying IPSO to PIPSO. Thus, correlation among variables is taken into consideration while selecting test functions. Functions can be categorized into two groups according to the relationship among their variables. Functions with uncorrelated variables are categorized into uncorrelated functions (UF), while functions with correlated variables fall

into correlated function (CF). Four test functions, including the Tripod, Rastrigin, Griewank and Rosenbrock functions, were used in Chapter 3 that were selected based on their popularity in the PSO community. Among them, there is only one UF, which is the Rastrigin function. To strike a balance between the two categories of functions, the tripod function is replaced by Ackley function that is also a widely used benchmark function in the PSO context. The mathematical expressions of the Rastrigin, Griewank and Rosenbrock are given in Equation (3.9), (3.10) and (3.11), respectively, and that of the Ackley function is shown below.

Ackley (UF): A well-known function that is very difficult because the "attraction basin" of the global minimum is quite narrow. The global optimum is achieved at $\mathbf{x}^* = (0, \ldots, 0)$, where $f(\mathbf{x}^*) = 0$.

$$f(\mathbf{x}) = 20 + e - 20\exp\Big(-\frac{1}{5}\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\Big) - \exp\Big(-\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)\Big), \qquad (4.7)$$

where $x_i \in [-30, 30]$.

Consequently, two of the test functions are UFs (Rastrigin and Ackley) and the other two are CFs (Griewank and Rosenbrock). Note that all functions are tested using 30-dimentional search spaces, for comparing the algorithms in a high dimensional environment.

All the present results are the mean values averaged over 60 independent runs to avoid experimental error. The comparison is conducted among PIPSO_CB, PIPSO_SB, IPSO, CPSO and the asynchronous version of the standard PSO (ASPSO). The PIPSO_CB and the PIPSO_SB correspond to PIPSO with the proposed fitness assignment methods for BBS, CB and SB, respectively. The IPSO has been investigated in chapter 3, based on which the PIPSO is developed. The CPSO has a similarity with PIPSO such that it also handles a problem dimension by dimension. In the ASPSO, the global best is updated after each particle position updating, which differentiates it from the original standard PSO. This asynchronous version

aims to improve the rate of convergence of the original one.

A t-test is completed to evaluate the significance of difference between two means, and p-value is reported for a certain claim. The significance level is set to 0.05. And $p = 0.05$ is used as the threshold for claiming significant difference, i.e., the difference is significant with at least 95% confidence level if its p-value obtained from t-test is less than 0.05.

A fair stopping criterion is necessary to compare the algorithms mentioned above. Using processor time as a stopping criterion may not be a good choice because this time is related to some other factors beside the efficacy of the used algorithm itself, such as software developing environment and programming skills. The number of function evaluations is used as the stopping criterion for its strong relationship with computational cost when the function complexity changes. All experiments are run for $6 \times 10_5$ function evaluations, which is set large enough for the algorithms to converge.

The program of ASPSO is downloaded from one of its designers' website (`http://www.engr.iupui.edu/~eberhart/`). The CPSO is programmed according to the description present in [54]. For heuristic optimization algorithms, the parameter values are generally chosen by trials. With this guideline, the parameters of PIPSO are tuned by a preprocessing procedure, which results in a satisfactory outcome. The values of the parameters of IPSO and ASPSO are inherited from Table 3.1. The parameters of CPSO are set to the values suggested by Bergh in [54]. The parameter configurations of the algorithms under comparison are shown in Table 4.2.

Table 4.2: Parameter configurations

Parameter Settings	PIPSO_CB/ PIPSO_SB	Number of cutting planes: $cp = 2$ Particle number per cutting plane: $cp\_pn = 10$ Iteration limitation per cutting plane: $cp\_maxiter = 15$ Inertia weight: $w = 0.729$ Acceleration constants: $c_1 = c_2 = 1.49445$ Mutation rate: $mutrate = 0.3$
	IPSO	Number of cutting planes: $cp = 2$ Particle number per cutting plane: $cp\_pn = 5$ Iteration limitation per cutting plane: $cp_maxiter = 10$ Inertia weight: $w = 0.729$ Acceleration constants: $c_1 = c_2 = 1.49445$
	CPSO	Swarm size: $s = 20$ Dynamic inertia weight: $w = \frac{(1-0.72)(max\_iter-iter)}{max\_iter} + 0.72$ Acceleration constants: $c_1 = c_2 = 1.494$
	ASPSO	Swarm size: $s = 20$ Dynamic inertia weight: $w = \frac{(0.9-0.4)(max\_iter-iter)}{max\_iter} + 0.4$ Acceleration constants: $c_1 = c_2 = 2$

Note: max_iter is the iteration limitation for each run and is a counter of iteration number.

4.4.3 Experimental Results and Analysis

In this section, firstly, the performance of the PIPSO on the selected benchmark problems, using the fitness assignment method of either CB or SB, is compared with that of IPSO, CPSO and ASPSO. Secondly, the robustness of the PIPSO is validated by comparing its performance with varying configurations.

Performance comparison

With the configurations shown in Table 4.2, the algorithms for comparison are run to solve the chosen benchmark problems. The experimental results are shown in

Table 4.3–Table 4.6.

Table 4.3: Performance comparison on Rastrigin function (UF)

	PIPSO_CB	PIPSO_SB	IPSO	CPSO	ASPSO
e	0.008458	0.006541	0.019513	0.024349	34.29886
(σ_e)	(0.007132)	(0.005917)	(0.003118)	(0.007981)	(10.45324)
γ	0.006530	0.005742	0.009917	0.011079	5.929987
(σ_γ)	(0.004942)	(0.004201)	(0.001033)	(0.002256)	(1.021667)
t	0.421443	0.419926	0.436163	0.416397	0.532656
(σ_t)	(0.033760)	(0.033801)	(0.035307)	(0.037153)	(0.050418)
η	75%	80%	53.3%	50%	0

Legends:
e: The error between the obtained optimal objective value and the objective value of the global optimum;
γ: The Euclidean distance between the found optimum and the global optimum;
t: The total computation time in second;
η: The rate of successful convergence;
(σ_e), (σ_γ) and (σ_t) are the standard deviations of e, γ and t, respectively.

In the Rastrigin problem, the performance of the algorithms can be partitioned into three groups. The PIPSO_CB and PIPSO_SB exhibit outstanding performance in terms of all the evaluation metrics, which indicates that the solutions found by these two algorithms are the closest to the global minimum both in the input space and in the output space. The IPSO and CPSO obtain similar values of e and γ, although those values of IPSO are always slightly better. The ASPSO never found the global optimum, whose performance is the worst. The t-tests on e show the significance of the performance advantage of PIPSO_SB (the performance winner) vs. IPSO, CPSO and ASPSO with $p < 0.01$. But the t-tests between PIPSO_CB and PIPSO_SB on e show no significant difference.

The results in the Ackley problem exhibits a pattern similar to that observed in the Rastrigin problem. The two PIPSO algorithms still take the lead. The IPSO slightly outperforms the CPSO, but stays behind the PIPSO algorithms with a large gap. The ASPSO never succeed in locating the global minimum. The t-tests

Table 4.4: Performance comparison on Ackley function (UF)

	PIPSO_CB	PIPSO_SB	IPSO	CPSO	ASPSO
e	0.004914	0.002965	1.481603	1.874619	7.392581
(σ_e)	(0.002019)	(0.001482)	(0.802811)	(0.54664)	(0.566542)
γ	0.003464	0.001068	1.195133	1.537481	6.092557
(σ_γ)	(0.001033)	(0.000935)	(0.673958)	(0.434902)	(0.673916)
t	0.601983	0.601298	0.608532	0.600933	0.720838
(σ_t)	(0.047465)	(0.049334)	(0.050147)	(0.052176)	(0.069929)
η	53.3%	58.3%	16.7%	11.7%	0

Legends: Refer to Table 4.3.

on are performed between PIPSO_SB versus each of the other four algorithms. Its performance is shown significantly different with IPSO, CPSO and ASPSO with $p < 0.01$, while not significantly different with PIPSO_CB.

Table 4.5: Performance comparison on Griewank function (CF)

	PIPSO_CB	PIPSO_SB	IPSO	CPSO	ASPSO
e	0.006713	0.005851	0.004012	0.022593	0.062159
(σ_e)	(0.005199)	(0.005116)	(0.005513)	(0.009061)	(0.025532)
γ	3.423612	2.739110	2.416988	5.074749	9.570022
(σ_γ)	(3.341098)	(3.810822)	(3.284386)	(3.481634)	(4.433809)
t	0.559355	0.556109	0.564931	0.552549	0.656367
(σ_t)	(0.041623)	(0.041274)	(0.043155)	(0.047586)	(0.056762)
η	50%	53.3%	61.7%	36.7%	10%

Legends: Refer to Table 4.3.

In the Griewank problem, an interesting result is observed that IPSO outperforms the two PIPSO algorithms, although the performance difference is small both in the input space and the output space. The three incremental algorithms obtain much smaller values of e and γ. The t-tests on e show no significant difference between IPSO and each of the two PIPSO algorithms. But the significance of the performance advantage on e of IPSO vs. CPSO and ASPSO is shown with $p < 0.01$.

Table 4.6: Performance comparison on Rosenbrock function (CF)

	PIPSO_CB	PIPSO_SB	IPSO	CPSO	ASPSO
e	1.078100	0.899057	0.933384	1.593373	2.072573
(σ_e)	(0.734983)	(0.734039)	(0.781011)	(0.816825)	(0.534971)
γ	1.073101	0.936957	1.052643	1.589539	2.159292
(σ_γ)	(0.641764)	(0.612402)	(0.623868)	(0.704199)	(0.408854)
t	0.447535	0.439211	0.460459	0.424125	0.476784
(σ_t)	(0.041027)	(0.040214)	(0.045188)	(0.047359)	(0.050418)
η	50%	53.3%	53.3%	50%	33.3%

Legends: Refer to Table 4.3.

In the Rosenbrock problem, the PIPSO_SB outperforms the other algorithms in terms of e and γ. The performance of IPSO is better than that of PIPSO_CB with weak dominance. The ASPSO still takes the last place. However, the advantage of the incremental algorithms shown in this problem is not as prominent as in other problems. The t-tests on e show significant difference of the performance advantage of PIPSO_SB vs. CPSO and ASPSO with $p < 0.01$, vs. PIPSO_CB with $p < 0.05$. No significant difference is shown between the performance of PIPSO_SB and IPSO.

With regard to t, it can be seen from Table 4.3 to Table 4.6 that all the algorithms took less than one second to solve the test problems. In all the problems, the ASPSO took the most computation time, the reason of which may be that the updating of each particle corresponding to a function evaluation in ASPSO involves all the dimensions, while only one dimension is involved in PIPSO, CPSO and SVOs of IPSO. The computation time of PIPSO, IPSO and CPSO is similar in all the problems. Precisely speaking, CPSO took less time than PIPSO, and PIPSO took less than IPSO. As stated above, the updating in PIPSO and IPSO only involves one dimension, but the number of dimensions involved in MVOs of IPSO is more than one that increases from two to d. The advantage of CPSO over PIPSO may be resulted from its smaller overhead. The overhead of PIPSO mainly comes from the maintenance of BBS. However, the difference among PIPSO, IPSO and CPSO

is not significant.

Sensitivity analysis

Say a complete run from SVO_1 to SVO_d is defined as an outer loop. The number of outer loops, denoted as *outlp*, is decided by the parameters of PIPSO given in Table 4.2, *cp*, *cp_pn* and *cp_maxiter*, with the total number of function evaluations, denoted by $fnevals$, being fixed. Their relationship can be described by the equation below.

$$fnevals = outlp \times cp \times (cp\_pn \times cp\_maxiter) \times d \tag{4.8}$$

Normally, swarm size impacts the performance of a PSO-based algorithm most. In PIPSO, there are two kinds of swarm size corresponding to two levels of searching. The lower-level searching is the searching performed in a cutting plane, where *cp_pn* and *cp_maxiter* are involved. So, *cp_pn* is regarded as the swarm size in this level. The upper-level searching is the movement of cutting planes, where *cp* can be regarded as the swarm size. Therefore, the changes of the performance of PIPSO on the benchmark problems will be observed as the value of *cp_pn* and *cp* independently varies. The results corresponding to the change of *cp_pn* and *cp* are shown in Table 4.7–Table 4.10 and Table 4.11–Table 4.14, respectively. On the other hand, the number of function evaluations used for searching in a cutting plane, i.e. $cp_maxiter$, does not have many choices. That is because the PSO converges very fast when it is used to search the cutting planes, namely dealing with 1D problems. Obviously, this number should be kept small (around 10) to avoid waste of computational cost. So, we did not study the sensitivity of PIPSO on $cp_maxiter$. What should be noted is that when *cp_pn* or *cp* changes, other parameters will be fixed except *outlp*, which is changed accordingly to keep $fneval$

unchanged. Additionally, the SB fitness assignment method is used in the following experiments.

Table 4.7: Sensitive Analysis on *cp_pn* (Rastrigin)

cp_pn	e (σ_e)	γ (σ_γ)	η
10	0.006541 (0.005917)	0.005742 (0.004201)	80%
15	0.006923 (0.005762)	0.005844 (0.004173)	76.7%
20	0.007606 (0.006465)	0.006061 (0.004317)	73.3%

Table 4.8: Sensitive Analysis on *cp_pn* (Ackley)

cp_pn	e (σ_e)	γ (σ_γ)	η
10	0.002965 (0.001482)	0.001068 (0.000935)	58.3%
15	0.003189 (0.001282)	0.001916 (0.001279)	56.7%
20	0.004103 (0.002290)	0.003009 (0.001410)	53.3%

Table 4.9: Sensitive Analysis on *cp_pn* (Griewank)

cp_pn	e (σ_e)	γ (σ_γ)	η
10	0.005851 (0.005116)	2.739110 (3.810822)	53.3%
15	0.006003 (0.005336)	2.987191 (2.988517)	50%
20	0.006474 (0.005445)	3.272211 (3.146640)	50%

It can be seen from Table 4.7–Table 4.10 that the performance of PIPSO de-

Table 4.10: Sensitive Analysis on *cp_pn* (Rosenbrock)

cp_pn	e (σ_e)	γ (σ_γ)	η
10	0.899057 (0.734039)	0.936957 (0.612402)	53.3%
15	1.672152 (0.838137)	1.64186 (0.820264)	40%
20	2.581321 (0.590099)	2.501961 (0.572133)	26.7%

creased when *cp_pn*, namely the swarm size in lower-level searching, increases. This trend can be observed in all the problems in terms of every metrics. The decrements in Rastrigin, Ackley and Griewank problems were not obvious, while in Rosenbrock problem were notable. For example, the decrements in terms of η in the former were less than 5%, while the values of η decreased by more than 10%–20% in the Rosenbrock problem. It can be inferred from the remarkable decrements observed in the Rosenbrock problem that the enhancement in searching in cutting planes may not help approach the global optimum and the reduced outer searching loops may be detrimental to the performance. This could be explained by the deceptively flat landscape of the Rosenbrock problem. Since the interceptive curves in the cutting planes are flat, 10 particles may be more than enough to solve this 1D problem. Thus, the increment of *cp_pn* may not be able to benefit PIPSO in approaching the global optimum. As mentioned before, *outlp* decreases as *cp_pn* increases in order to keep $fnevals$ be fixed. The deceptively flat landscape of the Rosenbrock function stops optimization algorithms to approach the global optimum. In other words, the algorithms tend to be unfortunately trapped in some point, the probability of jumping out of which may be greatly impacted by the value of *outlp*. When *outlp* is reduced, the probability may decrease, resulting in worse performance of PIPSO.

Table 4.11: Sensitive Analysis on *cp* (Rastrigin)

cp	e (σ_e)	γ (σ_γ)	η
2	0.006541 (0.005917)	0.005742 (0.004201)	80%
3	0.006568 (0.005931)	0.005801 (0.004447)	80%
4	0.008113 (0.006921)	0.006154 (0.004919)	75%

Table 4.12: Sensitive Analysis on *cp* (Ackley)

cp	e (σ_e)	γ (σ_γ)	η
2	0.002965 (0.001482)	0.001068 (0.000935)	58.3%
3	0.813295 (0.521988)	0.600153 (0.290099)	33.3%
4	1.419355 (0.801389)	1.120579 (0.592722)	16.7%

Table 4.13: Sensitive Analysis on *cp* (Griewank)

cp	e (σ_e)	γ (σ_γ)	η
2	0.005851 (0.005116)	2.739110 (3.810822)	53.3%
3	0.006445 (0.006183)	3.194186 (3.170952)	50%
4	0.006428 (0.006028)	3.206721 (3.189318)	50%

From Table 4.11–Table 4.14, it can be seen that except in Ackley problem the performance of PIPSO did not change much. In Rastrigin problem, the perfor-

Table 4.14: Sensitive Analysis on *cp* (Rosenbrock)

cp	e (σ_e)	γ (σ_γ)	η
2	0.899057 (0.734039)	0.936957 (0.612402)	53.3%
3	0.830246 (0.803784)	0.923028 (0.694966)	50%
4	0.951897 (0.721934)	1.025417 (0.615049)	48.3%

mance hardly decreased when *cp*, namely the swarm size in upper-level searching, increases from 2 to 3, and slightly decreased when *cp* consecutively increased to 4. The performance decrements in Ackley problem were quite notable in all the metrics, which may be caused by the narrow "attraction basin" of the global optimum. As mentioned before, *outlp* decreases as *cp* increases in order to keep $fnevals$ fixed. Normally, the increased number of cutting planes could compensate the reduced number of outer loops, resulting in stable performance. However, the mentioned narrow "attraction basin" results in tiny increase of the probability of approaching the global optimum by assigning more cutting planes. The cutting planes need to be adjusted according to the obtained information through a relatively long process, which needs large number of outer loops. This could explain why the performance of PIPSO is worse when *outlp* decreased. In Griewank problem, the performance of PIPSO slightly decreased when *cp* increases from 2 to 3, and hardly decreased when it increased from 3 to 4. In Rosenbrock problem, the performance in terms of η continuously decreased with slight difference when *cp* increased from 2 to 4. What should be noted is that when *cp* increased from 2 to 3, the performance in terms of e and γ slightly increased while η decreased. This shows that although the rate of convergence decreased, the average distance between the found optima and the true global optimum was reduced. This phenomenon suggests properly

increasing the cutting planes for solving functions with deceptively flat landscapes.

Both Chapter 3 and Chapter 4 focus on the incremental optimization in the input space. As the second topic of this thesis, the incremental optimization in the output space is studied in the next two chapters.

Chapter 5

Incremental Multi-Objective Optimization

After investigating the incremental optimization in the input space, this chapter studies the incremental optimization in the output space. The relationship between Pareto-optimal fronts before and after objective increment is theoretically analyzed based on the theorems about the corresponding relationship regarding Pareto sets increment given in [110]. An incremental model in the output space is backed with the proved relationship. Based on this model, a novel IMOPSO is implemented by applying the model to an MOPSO. Performance comparison between the MOPSO and the IMOPSO reveals the efficacy of the incremental model on several benchmark multi-objective optimization problems.

5.1 Effect of Objective Increment on Pareto Front

5.1.1 Definitions and Notations

With regard to the MOP (with n objectives) defined in Chapter 2, we suggest the following definitions to facilitate discussion:

1. If two decision vectors within the input space S give equal output in every objective, we say these two solutions are *phenotypically equal* to each other under the specified objective set. Similarly, if two decision vectors within S give different objective vectors, they are *phenotypically distinct* under the specified objective set.

2. A Pareto-optimal solution $\mathbf{x}$ is a *unique* Pareto-optimal solution (abbreviated as *unique P-solution*) if there is no other solution phenotypically equal to $\mathbf{x}$ within S.

3. A Pareto-optimal solution is a *non-unique* Pareto-optimal point (abbreviated as *non-unique P-solution*) if there is one or more solutions phenotypically equal to it within S.

4. A non-unique Pareto-optimal solution together with all the solutions phenotypically equal to it within S constitutes a *reduplicate Pareto-optimal group* (abbreviated as *reduplicate P-group*).

   Obviously, one unique P-solution corresponds to one Pareto-optimal point in the objective space, while all the solutions in a non-unique P-group correspond to the same Pareto-optimal point in the objective space.

5. A problem formed by a subset of the original objective set is called a *sub problem* of the original MOP.

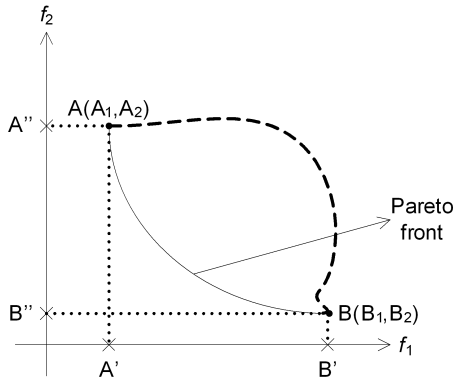


Figure 5.1: Illustration of m-proto by a 2D example

6. The orthographic projection of a Pareto front to an m-dimensional output space is called its *m-proto*, where $m = 1, 2, \ldots, n-1$. The 1-proto of the Pareto front of a 2D problem is shown in Figure 5.1 as an example. As shown in Figure 5.1, there are two 1-proto of the Pareto front stood by the arc AB, $A'B'$ and $A"B"$, corresponding to 1D spaces of f_1 and f_2, respectively.

In addition, the following notations are also used in this thesis:

1. P denotes the true Pareto front of an MOP;
2. P_m^o denotes an m-proto of P;
3. P_m denotes the Pareto front of a sub problem with m objectives.

5.1.2 Relationship between Pareto Fronts before and after Objective Increment

Theorem 5.1: *The Pareto front of a sub problem is covered by the m-proto of P in the corresponding subspace, i.e. $P_m \subseteq P_m^o$.*

Proof:

Without loss of the generality, we assume the sub problem with m objectives comprises of the first m objectives $\{f_1, f_2, \ldots, f_m\}$.

According to the definition of P_m^o, it is obtained from P by truncating $(n-m)$ elements from every point. For example, assume $F = \{f_1, f_2, f_3, f_4\}$, $m = 2$, and

$$\begin{aligned} f_1 &= |\sin\frac{\pi x}{2}| \\ f_2 &= |\cos\frac{\pi x}{2}| \\ f_3 &= (x-3)^2 \\ f_4 &= x^2 \end{aligned} \tag{5.1}$$

with the constraint: $x \in \{0, 1, \ldots, 100\}$.
In this example, P=$\{$(0,1,9,0),(1,0,4,1),(0,1,1,4),(1,0,0,9)$\}$, and P_2^o=$\{$(0,1),(1,0)$\}$. Please note that: 1) the duplicate members presenting after truncation geometrically mean that some points overlap after orthographic projection; 2) the values of the un-projected objectives remains the same without distortion.

Apagoge is used to prove the Theorem 5.1.

Assumption: There exists a point T which lies in P_m but not in P_m^o, i.e. $\{\exists T \mid T \in Pm, T \notin P_m^o\}$.

There are two situations:

1. T corresponds to a unique P-solution X in S.

   Inferred from the assumption, X will lose the dominance after objective increment, namely under F. In other words, it will be dominated by other solution, say Y, after objective increment. According to the definition of Pareto domination, there is:

$$f_i(Y) \leq f_i(X) \quad \forall i = 1, 2 \ldots, n \tag{5.2}$$

From Equation (5.2), it can be inferred that:

$$f_i(Y) \leq f_i(X) \quad \forall i = 1, 2 \ldots, m \tag{5.3}$$

Equation (5.3) means that X is either dominated by or phenotypically equal to Y before objective increment. If dominated, it contradicts with the premise that T, the output of X, lies in P_m. If phenotypically equal, it contradicts with the assumption that X is a unique P-solution.

A conclusion can be drawn from the proof in the first situation as a corollary of Theorem 5.1.

Corollary 5.1: The output of a unique P-solution found before objective increment will keep lying on the Pareto front after objective increment.

2. T corresponds to a reduplicate P-group R in S.

   Inferred from the assumption, all the non-unique P-solutions in R will lose the dominance after objective increment, namely under F. In other words, at least one of them will be dominated by other solution(s) outside R after objective increment. Say X ($X \in R$) is dominated by Y ($Y \notin R$) after the objective increment. According to the definition of Pareto dominance, there is:

$$f_i(Y) \leq f_i(X) \quad \forall i = 1, 2 \ldots, n \tag{5.4}$$

   From Equation(5.4), it can be inferred that:

$$f_i(Y) \leq f_i(X) \quad \forall i = 1, 2 \ldots, m \tag{5.5}$$

   Equation (5.5) means that X is dominated by or phenotypically equal to Y before objective increment. If dominated, it contradicts with the premise that T, the output of X, lies in P_m. If phenotypically equal, it contradicts with the assumption that $Y \notin R$.

A conclusion can be drawn from the proof in the second situation as another corollary of Theorem 5.1.

Corollary 5.2: At least the output of one non-unique P-solution in a replicate P-group found before objective increment will keep lying on the Pareto front after objective increment.

With the conclusion drawn from the two situations, Theorem 5.1 and Corollaries 5.1 and 5.2 are proved.

The theorem and corollaries proved above implied the feasibility to solve an MOP by taking objectives into consideration incrementally. The Pareto optimal solutions found under a sub problem are valuable after objective increment so that the Pareto front of a sub problem can be regarded as a base for sub problems after objective increment. Based on this rationale, an incremental model in the output space is built which will be described in the following section.

5.2 Incremental Model in the Output Space

Without special statement, the "incremental model" in this chapter means the incremental model in the output space.

As we all know, the performance of a certain tool will improve as its task gets easier. The development of the incremental model arose from this idea. It is assumed that the performance of a certain algorithm will improve, or at least will not degrade as the objective set gets smaller. The findings proved in Section 5.1 show the continuous existence between the Pareto-optimal sets before and after objective increment. As the solutions obtained under a small objective set contain better candidates, most of which are likely to stay after objective increment, they could assist the subsequent optimization under the incremented objective set. Thus, an

incremental approach can be more efficient. For illustration, the procedure of the proposed incremental model is shown in Figure 5.2.

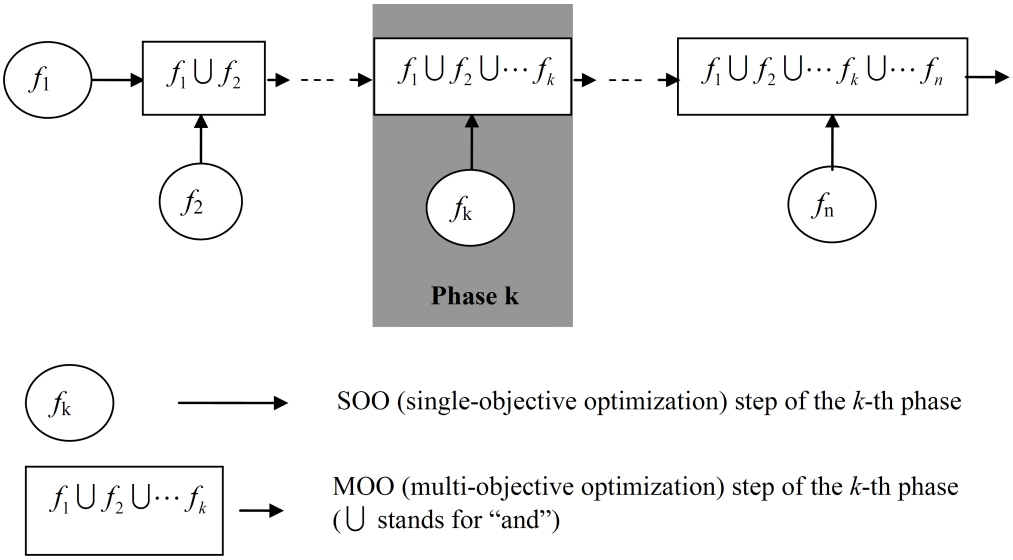


Figure 5.2: Incremental model in the output space
Remarks: What needs to be noted is that the positions of the first two functions are in equal status, thereby exchangeable.

As shown in Figure 5.2, the whole optimization is divided into as many phases as the objectives, and in each phase a new objective is considered. Each phase is composed of two stages: single-objective optimization (SOO) and multi-objective optimization (MOO). An independent population is evolved by SOO to optimize one specific objective. Next, the better-performing individuals obtained by SOO and last MOO are joined together by integration operation. The resulting population then becomes an initial multi-objective population, to which a multi-objective optimization based on the incremented objective set is applied. The pseudo-code of the incremental model is given below in Figure 5.3.

It may be argued that the cost of IMOO should be much higher than normal MOO algorithms. In fact, except the last MOO, the resources including population size and number of iterations are limited for the other MOOs. So the total cost of IMOO can be kept comparable with that of normal MOO algorithms. The resource limitation, such as population size and number of iterations, is the major reason

```
Randomly initialize a population;
Perform SOO on the population with the first objective;
Select s individuals to form the first multi-objective population (MOP);
Set k = 1;
Until (ALL THE OBJECTIVES HAVE BEEN EVOLVED) Do
{
        k = k + 1;
        Randomly initialize a population;
        Perform SOO on the population with the k th objective;
        Select s individuals to form the k th single objective population (SOP);
        Integrate the k th SOP with the (k − 1) th MOP to form a new population;
        Perform MOO based on the objective set from the first to the k th
objective;
        Select the best m individuals to form the k th MOP;
}
```

Figure 5.3: Pseudo-code of the incremental model in the output space

why objective ordering has impact on the performance of IMOGA, which will be discussed further in the next chapter.

5.3 PSO-based Incremental Optimization in the Output Space

5.3.1 Multi-Objective PSO (MOPSO)

In history, MOEAs have been widely used to solve MOPs. With the increasing belief in the merits of PSO such as easy implementation and fast convergence, more and more researchers recently work on employing PSO to solve the MOPs [10]. Among them, a state-of-the-art MOPSO with some small changes is used as the vehicle to investigate the effect of the incremental model. The pseudo code of this MOPSO is shown below in Figure 5.4, and a 2D example of the adaptive hypercube mechanism used is shown in Figure 5.5.

Initialize a swarm as well as the velocity and the personal best of each particle;
Store the non-dominated particles in the initial swarm in an external archive;
Generate hypercubes in the search space explored so far, and locate the particles using these hypercubes as a coordinate system where each particle's coordinates are defined according to the values of its objective functions;

Until (STOPPING CRITERIA ARE NOT REACHED) ***Do***
{
Select a global best for each particle:
1) roulette wheel selection on all the hypercubes (the fitness of a hypercube is associated with the number of particles lying in it),
2) random selection in the chosen hypercube;
Update the velocity and position for each particle;
Compare the newly generated particle with the original particle which acts as the local best): *if* it dominates the local best it will replace the local best, *if* neither of them is dominating choose the one with more better objectives to act as the local best;
Store all the non-dominated particles in the swarm in the archive;
Maintain the archive: eliminate the dominated particles and *if* the number of particles exceeds the archive size eliminate the particle(s) in the most crowded region(s);
}

Figure 5.4: Pseudo-code of the MOPSO

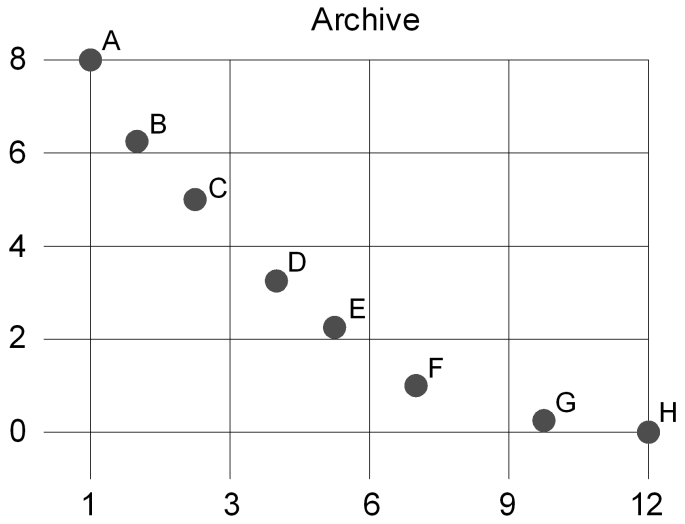


Figure 5.5: Graphical representation of hypercubes in 2D case

As shown in Figure 5.5, A to H are eight solutions contained in an archive. The space covered by these points are partitioned into grids. As described in Figure 5.4, the number contained in a grid is used to evaluate the fitness for choosing global-best, and the partition will be changed if the covered space increases as new member joins. The grids turn into cubes when the dimensionality of search space is three, and hyper-cubes when it is larger than three.

Different from MOEAs, MOPSOs does not employ any reproduction mecha-

nism which propagates the good values of variables over iterations. Instead, the population of MOPSO, named swarm, is expected to "fly" towards the Pareto front under selected guidance. Thus, a key issue of MOPSO is how to obtain correct and effective guidance.

In the MOPSO literature, most research work starts their search from scratch. Each non-dominated solution found during the search will be stored in an external archive and be deleted from the archive if it is dominated by some other solution found later on. This procedure is not very efficient in the sense that frequent replacement would take place in the archive so that large amount of dominance comparison has to be conducted.

Besides the high cost induced by the excessive dominance comparison, premature convergence may be another drawback of searching from scratch, especially in solving MOPs with more than two objectives. Usually the initial swarm of MOPSO will be randomly generated in order to distribute the particles uniformly in the input space. However, the corresponding distribution in the output space may not be uniform, especially in some high-dimensional output space. And PSO is famous for its fast convergence. So the particles may converge to a front near their initial positions, which may be far away from the true Pareto front.

In order to overcome the shortcomings of normal MOPSO mentioned above, the IMOPSO is proposed by applying the incremental model described in Section 5.2.

5.3.2 Incremental Multi-Objective PSO (IMOPSO)

By applying the incremental model to the MOPSO described in Section 5.3.1, the resulting IMOPSO will construct the final Pareto front dimension by dimension.

The Pareto front obtained in lower dimensional output space will be inherited to act as leader in the higher dimensional output space.

As shown in Figure 5.2, IMOPSO will divide the whole evolution into as many phases as the number of objectives. Each phase is composed of two stages. Firstly, in SOO, a swarm is optimized considering one specific objective. Secondly, in MOO, the better-performing particles obtained from stage one and the multi-objective swarm optimized in the last phase are joined together in stage two to become an initial swarm, to which a multi-objective optimization based on the incremented objective set is applied.

The algorithm works as follows (Assume there are n objectives and the Pareto archive size is also kept at N. Generally N is set equal to the number of solutions that the user desires.):

1. Set $k = 1$, where k is the phase sequence number. Generate a swarm and optimize it on the first objective for g_s iterations. After that, the p (The selection of parameter will be discussed later in Section 5.4.) fittest particles survive into M_1 (M_k represents the multi-objective inheritance from phase k). Phase 1, i.e. the initial phase, then ends.

2. Set $k = k + 1$. The next phase starts.

3. Generate a swarm and optimize it on the k-th objective. After that, the p fittest particles survive into S_k (S_k represents the single-objective inheritance from phase k.).

4. Randomly initialize a swarm for phase k. And generate the Pareto archive A_k for the MOO in this phase by the integration operation on S_k and M_{k-1}. The details of the integration operation will be given later in this chapter.

5. If the size of A_k is larger than N, select the N fittest particles. Note that if more than N particles have the same fitness, the ones in less crowded regions are selected. Then perform MOO on the initial swarm for g_M iterations using the MOPSO described in Section 5.3.1. Note also that only the first k objectives are considered in the selection.

   If $k < n$, set $g_M = g_{inter}$ (g_{inter} is the number of iterations for the intermediate phases). If $k = n$, the optimization goes on until the stopping criterion is met.

   After this procedure, M_k is obtained and phase k ends.

6. If $k < n$, go to step 2). If $k = n$, the whole optimization process finishes. The solutions in the Pareto archive are the solutions found.

From the procedure of IMOPSO described above, it can be seen that the integration operation plays an important role for IMOPSO. It combines the Pareto optimal solutions obtained in the low dimensional search spaces and propagates the useful information to higher dimensional search space.

The set of variables of the objective functions that a swarm is optimized for is called the relevant variables of this swarm. Accordingly, the irrelevant variables of this swarm are the whole variable set regarding all the objective functions excluding its relevant variable. The relevant variables of S_k is denoted as $V_{S,k}$, and those of M_k as $V_{M,k}$.

There are three types of integration operations, corresponding to three types of relationships between $V_{S,k}$ and $V_{M,k-1}$, as illustrated in Figure 5.6 to Figure 5.8:

1. If $V_{S,k} = V_{M,k-1}$, then the resulting swarm is the union of S_k and M_{k-1}.

2. If $V_{S,k} \bigcap V_{M,k-1} = \emptyset$, then each particle in S_k and each one in M_{k-1} when paired together will generate one offspring. The relevant variables in the offspring are copied from the correspondent parts in the parents.

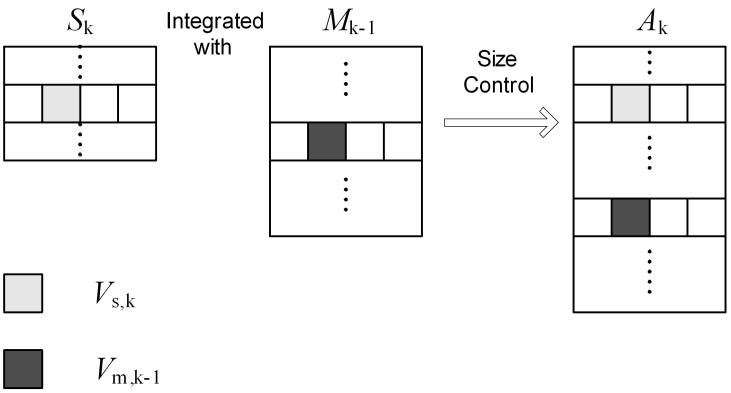


Figure 5.6: Integration operation ($V_{S,k} = V_{M,k-1}$)

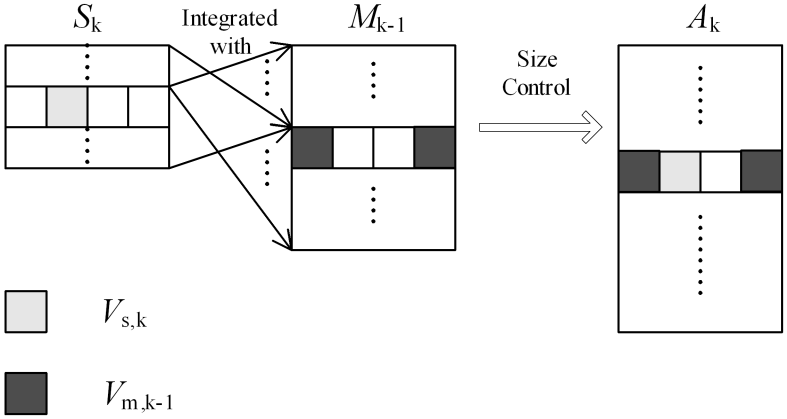


Figure 5.7: Integration operation ($V_{S,k} \bigcap V_{M,k-1} = \emptyset$)

3. If $V_{S,k} \neq V_{M,k-1}$ AND $V_{S,k} \bigcap V_{M,k-1} \neq \emptyset$, then each particle from S_k and M_{k-1} when paired together generate two offspring. In the offspring, the segments of those variables relevant to one parent only are copied from the corresponding parent. As to the variables relevant to both parents, each offspring will copy the segments in concern from one parent, respectively.

The idea of integration operation is to help combine useful information from S_k and M_{k-1} more efficiently. If $V_{S,k} = V_{M,k-1}$, it can be inferred, from the definition of relevant variables that $V_{M,k} = V_{S,k} \bigcup V_{M,k-1}$, that $V_{M,k} = V_{S,k} = V_{M,k-1}$. Under such circumstances, a resulting archive A_k, which is the union of S_k and M_{k-1}, can hold all the useful information. However, in an MOP, each objective may have different variable set. So it is possible that $V_{S,k} \neq V_{M,k-1}$, which results in either $V_{M,k} = V_{M,k-1}$ or $V_{M,k} = V_{S,k}$ or both. In these cases, if A_k is still simply the union

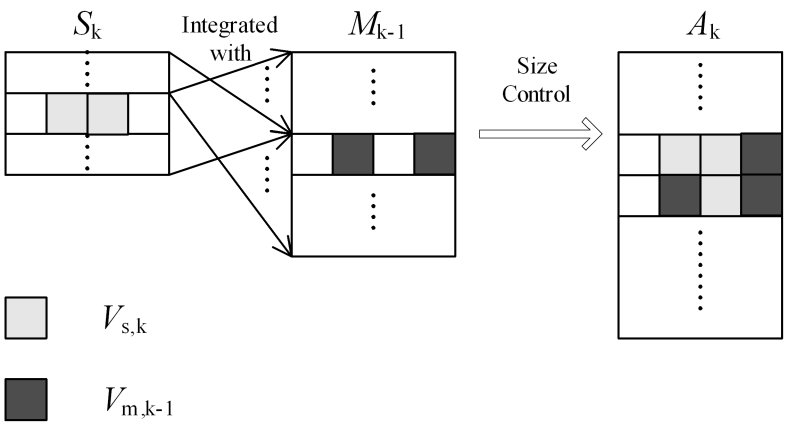


Figure 5.8: Integration operation ($V_{S,k} \neq V_{M,k-1}$ AND $V_{S,k} \bigcap V_{M,k-1} \neq \emptyset$)

of S_k and M_{k-1}, some problem will arise. Let us consider the situation $V_{M,k} = V_{S,k}$ as an example. In S_k, the segments of irrelevant variables are ignored by the optimization process. So, the segments corresponding to variables in $V_{S,k}$ but not in $V_{M,k}$ are randomly set without any optimization. The integration operation tries to set those segments more properly by copying from the corresponding segments in the particles from M_{k-1}, which have been optimized for the first (k-1) objectives.

Therefore, if $V_{S,k}$ and $V_{M,k-1}$ are different, each surviving particle from S_k and M_{k-1} are paired together as parents to generate particles for A_k. In particular, if $V_{S,k}$ and $V_{M,k-1}$ do not have any common variables, one offspring is enough to inherit information from the parent pair by copying from the corresponding variables in them. If $V_{S,k}$ and $V_{M,k-1}$ are not the same but have one or more common variables, only one offspring is insufficient to preserve the information. So, each pair of parents generates two offspring. Each offspring takes the value(s) of the common variable(s) from one parent, respectively, and both of them inherit the value(s) of the unique variable(s) from the two parents.

5.4 Experiments

Experiments for investigating the performance and properties of IMOPSO have been conducted and the results are reported in this section.

5.4.1 Performance Evaluation Metrics

Indicated by Zitzler [88], multi-objective optimization is quite different from single objective optimization in that there is more than one goal:

1. Minimize the distance of the Pareto front produced by our algorithm with respect to the true Pareto front.
2. Maximize the spread of solutions found, so that we can have a good (in most cases uniform) distribution of the solutions.
3. Maximize the extent of the non-dominated front obtained.

Therefore, the performance evaluation of multi-objective optimization is a non-trivial task. A lot of metrics have been proposed [104–108, 110]. In this thesis, the following metrics are used, corresponding to the goals mentioned above:

1. GD (indicates the closeness of the solutions to the real Pareto-front) and σ_{GD} (indicates how uniformly they approach the front) are metrics describing the solutions' convergence degree. To compute them, first find a set of true Pareto-optimal points uniformly spaced in the objective space. Then for each solution, we compute its minimum Euclidean distance to the true Pareto-optimal points. The average of these distances is GD, and the variance of the distances is σ_{GD}.

2. The coverage of the solutions is described by the metric η, according to he volume-based scaling-independent S metric and D metric proposed by Zitzler [110] with some slight modification, we define:

   E is the Pareto front found by IMOPSO algorithms;

   T is the true Pareto front found by a brutal-force method;

   $V = S(T)$, which is the hypervolume of the objective space dominated by the true Pareto front;

   $\alpha = D(T, T) = S(T + E) - S(E)$, which is the hypervolume of the objective space dominated by the true Pareto font but not by the found one;

   Here, V is set as the reference volume and the coverage metric $\eta = \frac{\alpha}{V}$, which aims to measure the successfully covered objective space by the IMOPSO algorithms. If η is close to 0, the solutions can be regarded as just covering the majority of the Pareto front.

3. SP measures how uniform the solutions spread. To compute SP, for every solution, find out its minimum normal Euclidean distance (denoted as N_E) to the other solutions. The variance of these distances is SP. The definition is given below. It is designed in such a way to avoid bias among objectives whose extents may be quite different. The smaller SP is, the better they are distributed.

$$N_E(a, b) = \sqrt{\sum_{k=1} n\Big(\frac{f_k(a) - f_k(b)}{t_k^{max} - t_k^{min}}\Big)^2}, \tag{5.6}$$

   where, a and b are points in Pareto front, $f(\cdot)$ is the objective value of the found Pareto front in the kth objective, and $t_k^{max} - t_k^{min}$ is the extent of true Pareto front in the kth objective. n is the number of objectives.

5.4.2 Experimental Scheme

To show the efficacy of the incremental model, IMOPSO is compared with MOPSO. The IMOPSO is also compared with three state-of-the-art MOGAs, including IMOGA, SPEA and NSGA-II, to further show its advantage. These MOGAs are selected for their distinctive properties. The IMOGA can be seen as an incremental algorithm resulted from applying the incremental model to normal MOGA [110]. In SPEA, an external population is maintained to store the non-dominated solutions discovered so far and this external population participates in all genetic operations [94, 95]. In NSGA-II, no repository is used, but a sorting mechanism is applied instead to maintain the dominance structure of the population [35]. In addition, a variation of the IMOPSO participates in the comparison, which performs integration at the middle of MOOs instead of at the beginning. It is called IMOPSO-II. This variation, named IMOPSO-II, aims to strike a balance between exploration and exploitation. Since the solutions inherited from SOOs will dominate most of the initial swarm in MOO, the diversity of the Pareto archive will be small. In other words, if they are introduced at the beginning of MOO, the MOO tends to converge quickly towards the inherited Pareto front, which may result in premature convergence. Theoretically, there are lots of choices of the timing, such as one fourth, one third etc. However, any choice will not be the best for all the problems. The half mechanism is used universally as a tradeoff according to a preprocessing procedure described in Appendix A.

Moreover, comparisons are performed to investigate the impact of the time of integration on the performance of IMOPSO. The whole MOO is divided into several periods, the inherited solutions from SOO and previous MOO can be integrated into the archive at the beginning of different periods, resulting in different results.

For fairness, all the comparison is based on the same number of function evaluations, denoted by $fnevals$. What should be noted is that each evaluation of an

individual objective function counts for $fnevals$, rather than count the number of the evaluations of all the objective functions in a MOP, as only one objective function is evaluated in SOOs of IMOPSO.

The experiments have been performed on several benchmark problems selected for their different levels of difficulties, which make the convergence complicated. These problems are divided into three groups according to their number of objectives, 2-, 3- and 4-objective problem. All the results are the average of 30 independent runs. In each run a different random sequence is used. The source codes of SPEA and NSGA-II were obtained from the Evolutionary Multi-Objective Optimization (EMOO) repository (`http://www.lania.mx/~ccoello/EMOO/`), and those of IMOGA and MOPSO were obtained from their developers [10, 118].

The following parameters were set according to their original papers and kept the same in all the experiments:

1. For all the MOGAs, each decision variable is encoded with 30bits, the mutation rate for each decision variable is $\frac{1}{d}$ and for each bit it is $\frac{1}{l}$, where d is the number of decision variables and l is the length of the chromosomes.

2. For IMOGA, each SOO is run with a population size of 100 for 10 iterations.

3. For SPEA, the ratio of population size to the external population is 4:1.

4. For NSGA-II, the crossover probability is 0.9, and the distribution indices for crossover and mutation are $\eta_c = 20$ and $\eta_m = 20$.

5. For MOPSO/IMOPSO, the mutation rate is 0.5, and the number of grids on each dimension for adaptively generating hypercubes is 30.

6. For IMOPSO, each SOO runs with a population size of 10 for 100 iterations, namely $g_s = 100$.

5.4.3 Results and Analysis

Three groups of benchmark MOPs are used to test the IMOPSO below.

2-objective MOPs

Four 2-objective benchmark MOPs are chosen from EMOO literature for their different characteristics of difficulties [114]. These problems, with their number of decision variables denoted as d, their variable ranges, their Pareto-optimal fronts and the feature of their fronts are given in Table 5.1. Table 5.1: 2-objective problems used to test IMOPSO

Table 5.1: 2-objective problems used to test IMOPSO

Prob.	d	Variable range	Objective functions	Pareto-optim front
FON	3	[-4,4]	$f_1 = 1 - \exp(-\sum_{i=1}^{n}(x_i - \frac{1}{\sqrt{3}})^2)$ $f_2 = 1 - \exp(-\sum_{i=1}^{n}(x_i + \frac{1}{\sqrt{3}})^2)$	Non-convex
KUR	3	[-5,5]	$f_1 = \sum_{i=1}^{n-1}(-10\exp(-0.2\sqrt{x_i^2 + x_{i+1}^2}))$ $f_2 = \sum_{i=1} n(\lvert x_i\rvert^{0.8} + 5\sin x_i^3)$	Non-convex a Disconnecte
ZDT1	30	[0,1]	$f_1 = x_1$ $f_2 = g(x)[1 - \sqrt{\frac{x_1}{g(x)}}]$ $g(x) = 1 + 9\frac{\sum_{i=2}^{n}}{n-1}$	Convex
ZDT3	30	[0,1]	$f_1 = x_1$ $f_2 = g(x)[1 - \sqrt{\frac{x_1}{g(x)}} - \frac{x_1}{g(x)}\sin(10\pi x_1)]$ $g(x) = 1 + 9\frac{\sum_{i=2}^{n}}{n-1}$	Convex and Disconnect

For this group of problems, the total number of function evaluations was set at

2×10^4 as suggested in [114]. The specific configurations for the algorithms under comparison are listed below, which are suggested by their original papers:

- The IMOGA used a population size of 100 for its MOO stage.
- The SPEA was run with a population size of 80 and an external population size of 20.
- The NSGA-II was run with a population size of 100.
- The MOPSO/IMOPSO used a population of 100 particles and an archive of 100 particles.

The comparison results are shown in Table 5.2 to Table 5.5.

Table 5.2: Comparison of results on FON

Algorithm	GD	σ_{GD}	η	SP
IMOPSO	0.001233	2.26e-06	0.015456	2.02e-06
IMOPSO-II	0.001018	1.99e-06	0.013658	1.77e-06
MOPSO	0.002064	2.68e-06	0.020803	1.93e-06
IMOGA	0.001871	2.06e-06	0.019416	4.73e-06
SPEA	0.006976	5.36e-05	0.132382	8.94e-04
NSGA-II	0.002231	4.32e-06	0.020983	2.86e-06

In the above five problems, the performance of IMOPSO algorithms is better than both the MOPSO and the IMOGA/MOGAs in all the metrics. The performance dominance of the IMOPSO over the IMOGA/MOGAs is much more significant in ZDT1 and ZDT3 than in FON and KUR. An interesting observation is that the MOPSO outperformed IMOGA/MOGAs a lot in ZDT1 and ZDT3, while its performance is slightly surpassed by IMOGA in FON and KUR except the SP metric. This finding points out consistency of the trend of performance change

Table 5.3: Comparison of results on KUR

Algorithm	GD	σ_{GD}	η	SP
IMOPSO	0.036031	0.001813	0.008372	2.91e-06
IMOPSO-II	0.032856	0.001004	0.007644	1.95e-06
MOPSO	0.041895	0.001957	0.012407	3.72e-06
IMOGA	0.040037	0.001844	0.010553	3.93e-06
SPEA	0.142397	0.018913	0.149853	1.44e-04
NSGA-II	0.052136	0.002047	0.014214	3.05e-06

Table 5.4: Comparison of results on ZDT1

Algorithm	GD	σ_{GD}	η	SP
IMOPSO	0.000987	8.77e-07	0.001369	2.34e-07
IMOPSO-II	0.000796	8.43e-07	0.001192	2.31e-07
MOPSO	0.001811	1.02e-06	0.002793	2.88e-07
IMOGA	0.012834	6.43e-06	0.003273	3.01e-07
SPEA	0.030614	6.25e-04	0.070276	9.28e-05
NSGA-II	0.015547	9.82e-06	0.007641	3.28e-06

Table 5.5: Comparison of results on ZDT3

Algorithm	GD	σ_{GD}	η	SP
IMOPSO	0.007444	1.57e-06	0.008983	3.11e-07
IMOPSO-II	0.005762	1.23e-06	0.008454	3.04e-07
MOPSO	0.009921	2.02e-06	0.010231	3.35e-07
IMOGA	0.023406	2.15e-06	0.016513	2.81e-06
SPEA	0.047747	6.63e-06	0.069500	9.73e-05
NSGA-II	0.028188	2.75e-06	0.028451	6.69e-07

between the IMOPSO and MOPSO as shown in Figure 5.9, which may prove the hypothesis that the incremental model can benefit a multi-objective optimization algorithm, making it achieve Pareto fronts closer to the true Pareto front and more

uniformly distributed. The exception in SP metric may result from the use of hypercubes in the MOPSO, which makes the Pareto-optimal solutions in the archive of the MOPSO spread uniformly. Another important observation is that in all the problems the performance of the IMOPSO-II was better than that of the IMOPSO, which suggest introducing the inherited solutions from a SOO at the middle of a MOO procedure rather than at the very beginning is better. The reason may be that the solutions inherited from SOOs will dominate most of the initial swarm in MOO, which would slow down the expansion of Pareto archive, thereby resulting in global best with small diversity. Thus, late introduction of the inherited solutions could strike a balance between exploration and exploitation.

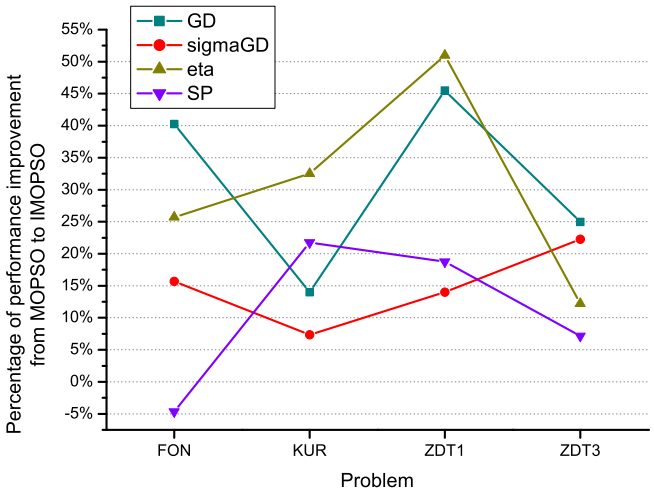


Figure 5.9: Percentage of the performance improvement from MOPSO to IMOPSO

A 3-objective MOP: VLMOP3

Veldhuizen and Lamont's MOP3 [115] is a 3-objective problem with 2 decision variables:

$$
\begin{aligned}
f_1 &= 0.5(x_1^2 + x_2^2) + \sin(x_1^2 + x_2^2) \\
f_2 &= \frac{(3x_1 - 2x_2 + 4)^2}{8} + \frac{(x_1 - x_2 + 1)^2}{27} + 15, \quad x_1, x_2 \in [-3, 3]. \\
f_3 &= \frac{1}{x_1^2 + x_2^2 + 1} - 1.1\exp(-x_1^2 - x_2^2)
\end{aligned} \tag{5.7}
$$

The VLMOP3 has a disconnected Pareto optimal set, and its Pareto front is a curve "following a convoluted path through objective space".

With the number of objectives increased to three, the total number of function evaluations for this problem was increased to 5.4×10^4 so that the algorithms can converge. The specific configurations for the algorithms under comparison are listed below, which are suggested by their original papers:

- The IMOGA used a population size of 120 for its MOO stage.
- The SPEA was run with a population size of 96 and an external population size of 24.
- The NSGA-II was run with a population size of 120.
- The MOPSO/IMOPSO used a population of 120 particles and an archive of 120 particles.

The comparison results are shown in Table 5.6.

In the VLMOP3 problem, MOPSO showed worse performance compared to IMOGA, although it outperformed SPEA and NSGA-II. However, with the improvement from MOPSO, IMOPSO/IMOPSO-II outperformed IMOGA in all the metrics considered as shown in Figure 5.10.

Table 5.6: Comparison of results on VLMOP3

Algorithm	GD	σ_{GD}	η	SP
IMOPSO	0.197892	0.011632	0.028577	1.2e-07
IMOPSO-II	0.150788	0.007061	0.021367	9.1e-08
MOPSO	0.292087	0.018512	0.050001	2.2e-07
IMOGA	0.231693	0.013474	0.042909	1.5e-07
SPEA	0.488524	0.021984	0.096928	4.1e-07
NSGA-II	0.301706	0.018524	0.052626	2.2e-07

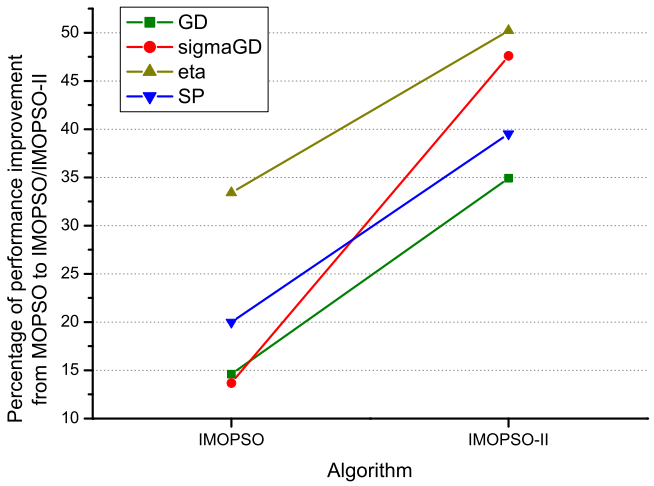


Figure 5.10: Percentage of the performance improvement from MOPSO to IMOPSO/IMOPSO-II

Case Study

For investigating the performance of the IMOPSO on real-world problems, a real-world example on optimization of polymer extrusion is studied.

Extrusion is a processing techniques used by companies producing plastic parts. In a typical single screw extruder, as shown in Figure 5.11, an Archimedes-type screw rotates at a given frequency inside a heated barrel. The aim of the process is to receive the solid pellets at the hopper, melt and mix the material, and pump it at a constant rate through the die, which is coupled to the other end of the barrel,

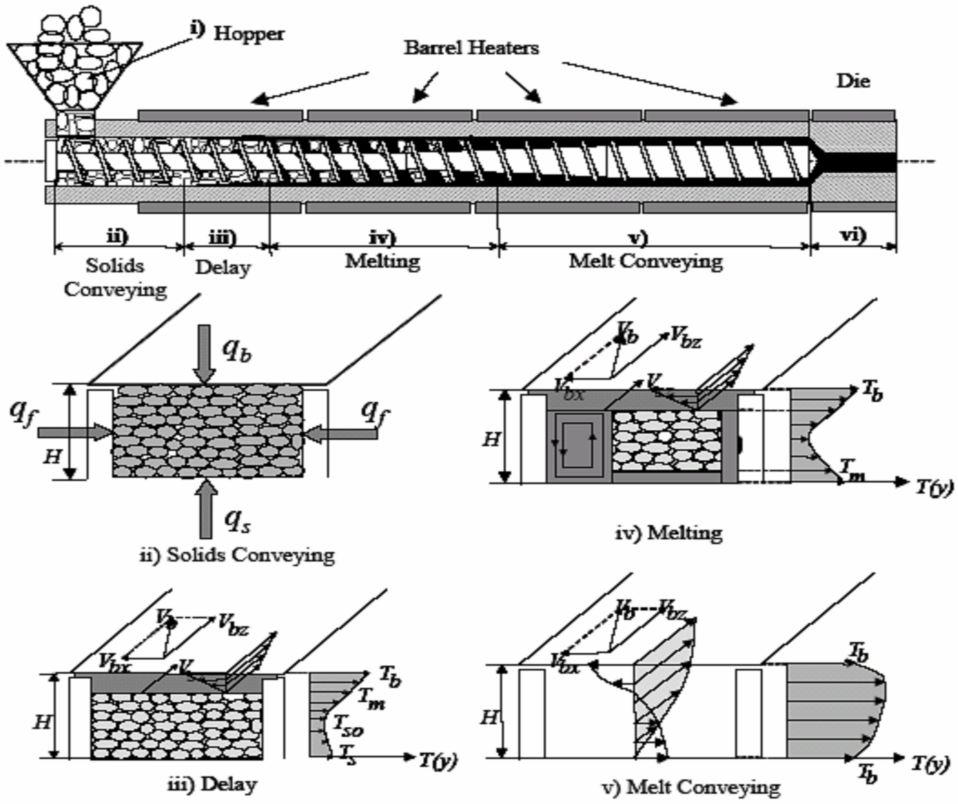


Figure 5.11: Processing sequence inside a polymer extruder [116]

in order to produce an extrudate with a prescribed cross-section.

The optimization for parameters of the extruder was studied as a MOP with 4 decision variables and 5 objectives in [116]. A modeling routine "extr.exe" of the extrusion process in binary code with two data files "extreq" and "extrmat" are available through the Internet (`www.dep.uminho.pt/pp/index.php3?gaspar@dep.uminho.pt`), which are for mapping variable vectors to objective points. To use this routine, a file "extr.var" should be created by an EMOO algorithm to pass the variable values of a solution to the modeling routine. After running the routine, another file "extr.cri" will be created giving the objective values of that solution, which should be passed to the EMOO algorithm.

This MOP is used to test IMOPSO in this thesis, whose detailed description regarding the variables and objectives are shown in Table 5.7 and Table 5.8, respectively.

Table 5.7: Variables of extruder optimization problem

Operating condition (Decision variable)	**Min**	**Max**
Screw speed N (rpm)	10	50
Barrel temperature profile $T1(^oC)$	150	210
Barrel temperature profile $T2(^oC)$	150	210
Barrel temperature profile $T3(^oC)$	150	210

Table 5.8: Objectives of extruder optimization problem

Criterion (Objective)	**Aim**	**Min**	**Max**
Mass output $Q(kg/hr)$	Maximize	1	10
Length for melting $Z_T(m)$	Minimize	0.2	0.9
Melt temperature at die exit $T_{exit}(^oC)$	Minimize	150	210
Power consumption $Power(W)$	Minimize	0	9200
Mixing quality $WATS$	Maximize	0	1300

For this real-world problem, the total number of function evaluations is up to four, which requires more function evaluations than previous problems to make problem solvers converge. So, the number was set at 9×10^4. The specific configurations for the algorithms under comparison are listed below, which are set according to the suggestions given in their original papers:

- The IMOGA uses a population size of 150 for its MOO stage and evolved 5 generation for the MOOs in intermediate phases.
- The SPEA is run with a population size of 120 and an external population size of 30.
- The NSGA-II is run with a population size of 150.
- The MOPSO/IMOPSO use a population of 150 particles and an archive of 150 particles.

Table 5.9: Comparison results on the extruder optimization problem

Algorithm	GD	σ_{GD}	η	SP
IMOPSO	1.465759	0.157950	0.129613	7.13e-06
IMOPSO-II	0.931497	0.072128	0.072966	5.14e-06
MOPSO	1.855142	0.214638	0.180652	9.66e-06
IMOGA	2.385149	0.209906	0.260654	7.01e-06
SPEA	4.702481	1.089589	0.454694	1.54e-04
NSGA-II	3.575186	0.347949	0.318738	1.12e-05

Generally, the findings in this real-world extruder optimization problem are consistent with what were observed from the benchmark problems shown earlier. Firstly, IMOPSO algorithms obtained a better performance in all the metrics than all the other algorithms under comparison, when MOPSO outperformed the IMOGA/MOGAs. Secondly, IMOPSO-II outperformed IMOPSO. What should be noted is that the superiority of the IMOPSO-II over IMOPSO was generally greater in this problem than in the numerical benchmark problems tested above, especially obvious compared to the 2-objective problems, which can be seen from Figure 12. This finding could be explained by the increase of the number of objectives, thereby the increase of the number of phases. As stated before, early introduction may restrict exploration, resulting in premature convergence. Since the rationale behind incremental multi-objective optimization algorithms is to construct Pareto fronts in higher dimensional spaces based on Pareto fronts obtained in lower dimensional spaces, the error in earlier phases may be inherited and amplified in subsequent phases. Thus, the probability and intensity of being impacted by the premature convergence in some phases may increase as the number of phases increases. That could explain why the performance difference between the IMOPSO and IMOPSO-II is more significant in this 5-objective problem.

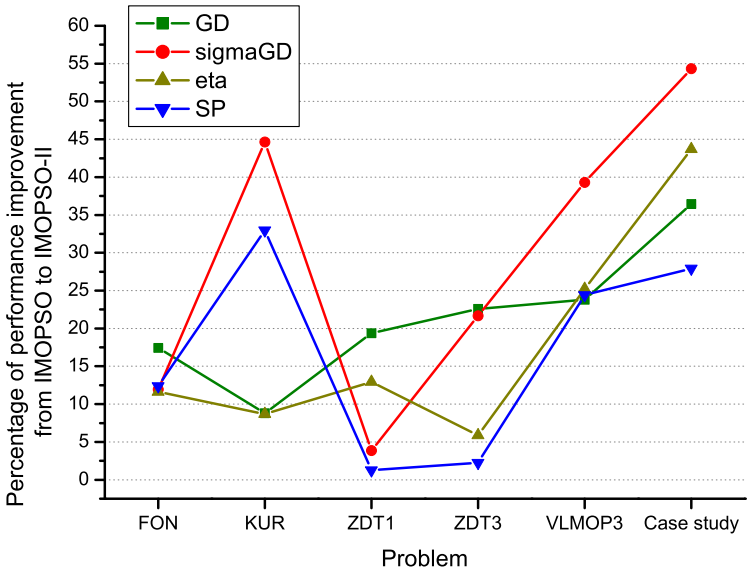


Figure 5.12: Percentage of performance improvement from IMOPSO to IMOPSO-II

5.4.4 Discussion

The success of a swarm is attributed to three fundamental processes: identification of a set of leaders, selection of a leader for information acquisition, and finally a meaningful information transfer scheme. The merits of IMOPSO could be seen from the three aspects.

First of all, the incremental optimization facilitates the identification of leaders. Normal MOPSOs consider all the objectives as a whole from the very beginning of optimization, and their search starts from scratch. That means lots of temporary dominating solutions will be stored into the Pareto-optimal archive, and be eliminated later when they are dominated by any newly found solutions. The large amount of dominance comparison may result in high computational cost, especially for problems with large objective number. In contrast, IMOPSO applies PSO-based incremental multi-objective optimization, which optimizes the objectives one by one. The results of SOO act as the leaders for the following MOO. It

has been proved that the global optimum of single objective is certainly a Pareto-optimal solution and even the local optimum of single objective corresponds to a point close to the Pareto front. So, by inheriting the results of SOO, the Pareto-optimal archive can be filled with some solutions of good quality at the beginning or in the middle of the MOO, which can reduce the number of temporary points entering the archive. Since the time PSO needs to solve single objective problems is little, it is reasonable to hypothesize that the time saved by inheritance will be more than enough to compensate the cost for SOO.

Secondly, as for the selection of personal best, most of current MOPSOs adopt random selection when a particle and its personal best cannot dominate each other. In IMOPSO, a partial dominance based selection is used. In other words, when the situation above happens, the probabilities of being selected as the personal best will be decided according to the number of objectives they outperform each other. This mechanism can slightly improve the performance of MOPSO, which was observed when the algorithm was implemented. Since the difference is very small, it is not shown in tables for avoiding confusion.

Thirdly, the IMOPSO constructs Pareto fronts in higher dimensional spaces based on Pareto fronts obtained in lower dimensional spaces. This may lead to fast convergence. In addition, to prevent premature convergence, the time of inheritance can be delayed to the middle of MOO. As stated earlier, the choice of introducing the inherited solutions is not exclusive. There does not exit a choice which is the best for all the problems. The half mechanism is used as a rule-of-thumb.

Moreover, the number of conflicting objectives of a real-world MOP is normally less than 5. Therefore, most of the research works on multi-objective optimization focus on 2-objective MOPs. In this thesis, the test cases include 2-objective, 3-objective and 5-objective problems. The performance change with the increase of the number of objectives is investigated. The experimental results show that

the larger the number of objectives is, the more the incremental algorithms outperform the non-incremental algorithms. In addition, the 5-objective problem is actually a real-world MOP. The experimental results of this case study show that the incremental model not only works for solving benchmark problems, but also for real-world applications.

In this chapter, the proposed model for incremental multi-objective optimization has been shown useful on the benchmark problems tested, with MOPSO being used as a vehicle. When using this model, the objectives of an MOP are intuitively handled by their original order. However, with an in-depth consideration, the ordering issue arises, which will be addressed in the next chapter.

Chapter 6

Ordered Incremental Multi-Objective Optimization

In the last chapter, the incremental multi-objective optimization (IMOO) has been proposed based on theoretical analysis and the IMOPSO has been proved effective experimentally. However, there is an open question remaining: how to decide the order of the objectives handled by IMOO? Due to the incremental nature, it is found that the ordering of objectives would impact the performance of IMOO, measured by the hyper-volume metrics. This chapter aims at solving this problem. We suggest that ordering can be determined by ranking pairs of objectives according to their level of conflict, followed by a recursive objective ordering approach that finds the optimal objective order to minimize the cumulative level of conflict. The experimental results from four multi-objective optimization problems show that the proposed objective ordering procedure can help IMOO reach its best performance.

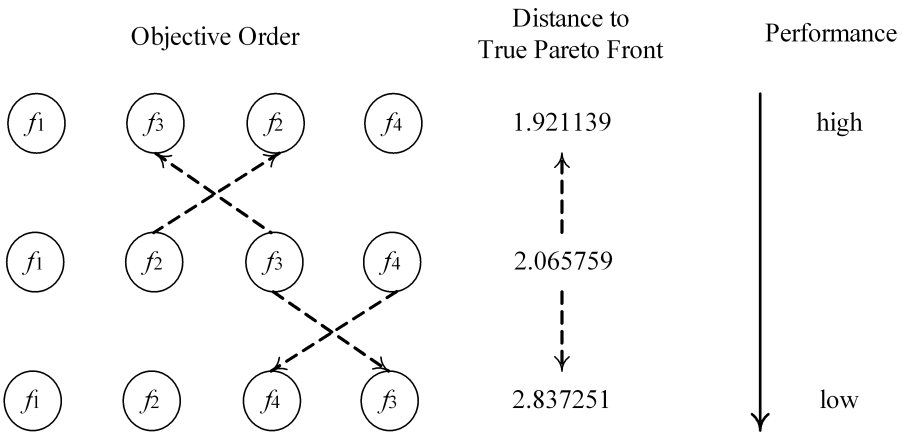


Figure 6.1: Performance fluctuation of IMOO with objective order changing

6.1 Motivation and Methodology

6.1.1 Motivation of Objective Ordering for IMOO

The investigation on IMOO has analyzed, proved and shown the effect of the incremental model on standard multi-objective optimization algorithm. So far, all the experiments handled the objective sets in their original order without considering the ordering issue. According to our subsequent study, the performance of IMOO fluctuates a lot with changes in objective order, though it outperforms the non-incremental algorithm with all the possible objective orders. Generally, the original objective order does not result in the best performance. This can be seen from the following example, which is a 4-objective problem solved by IMOO.

As shown in Figure 6.1, the performance of IMOO varied as the objective order changes, which is measured by the distance between the found solutions and the true Pareto front. When the objective functions of this problem were evolved in their original order, the distance was 2.065759. If the positions of the last two objective functions were exchanged, the distance was increased to 2.837251, i.e. the performance dropped by 37.35%. Whereas, if the middle two functions were exchanged, the distance was decreased to 1.921137, i.e. the performance was

improved by 7%. This finding suggests investigating the issue of objective ordering. Therefore, the aims of research are to evaluate the effect of objective ordering in incremental multi-objective problem solving, and to find out metrics for ranking the objective ordering so that the best objective order can be found.

To shed light on the possible reason behind the performance fluctuation with the change of objective ordering, the ordering effect on a minimization problem with three conflicting objectives is visualized here. As shown in Figure 6.2, it's assumed that:

1. The three center points O_1, O_2 and O_3 are the global optima for the three objective functions respectively;

2. Each circle represents a local optima circle, on which all the points have the same objective values. And the bigger the circles the larger the objective values. (The local optima circles could be expanded to the whole feasible input space, not limited to those shown in the figure.)

3. The density of the local optima circles implies the difficulty of an objective function. And the distance between two center points implies the number of Pareto-optimal solutions of the two corresponding objective functions.

It can be seen from Figure 6.2 that the Pareto set of $f_1 \bigcup f_2$, $f_1 \bigcup f_3$ and $f_2 \bigcup f_3$, should be O_1O_2, O_1O_3 and O_2O_3, respectively. Assume O_1O_2, O_1O_3 and O_2O_3 have equal lengths, the numbers of Pareto-optimal solutions for the three objective pairs would be the same if the resolution is fixed. Also, it can be seen that the densities of the local optima circles surrounding O_1 and O_3 are the smallest and the biggest respectively, which means that SOO_1 the will most likely get O_1 while SOO_3 the will be most likely trapped on local optima circles. Note that SOO_i stands for the ith SOO stage, i.e. the SOO stage in the ith phase of the procedure of IMOO, and

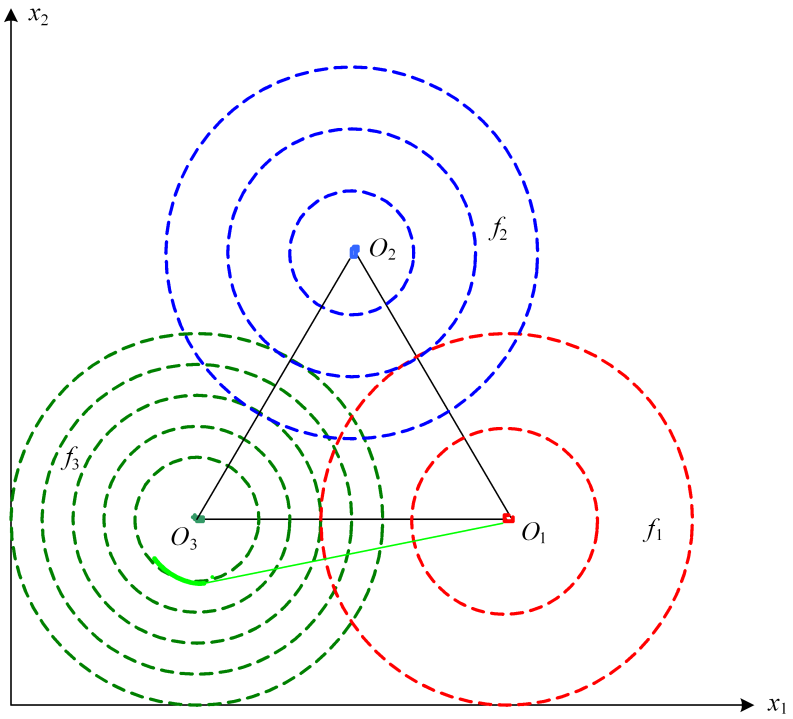


Figure 6.2: Three conflicting objectives with different difficulties

MOO_j stands for the jth MOO stage, i.e. the MOO stage in the $(j+1)$th phase, where $i = 1, 2 \ldots, n$ and $j = 1, 2 \ldots, n-1$.

Therefore, if f_1 and f_2 are the two objectives firstly optimized, it can be assumed that O_1 and O_2 are found by SOO_1 and SOO_2 respectively. Through integration, O_1 and O_2 can be inherited into MOO_1, which evolves $f_1 \bigcup f_2$ mainly by using local searching (mutation), i.e. $(x_1(t+1), x_2(t+1)) = (x_1(t)+\Delta_1, x_2(t)+\Delta_2)$. Thus, under the guide of O_1 and O_2, the Pareto set O_1O_2 will be found most likely. In contrast, if f_1 and f_3 are the two objectives firstly optimized, as shown in Figure 6.2 the light green arc may be found by SOO_3, as SOO_3 is very likely to be trapped. Then, under the guide of this arc and O_1, the light green line might be found as the Pareto set by MOO_1. Also, this biased Pareto set would make false guidance in the following MOO_2.

Similarly, the effect of the number of Pareto-optimal solutions can be illustrated by Figure 6.3, in which the three objective functions have the same possibility of being trapped on local optima circles, while the numbers of Pareto-optimal

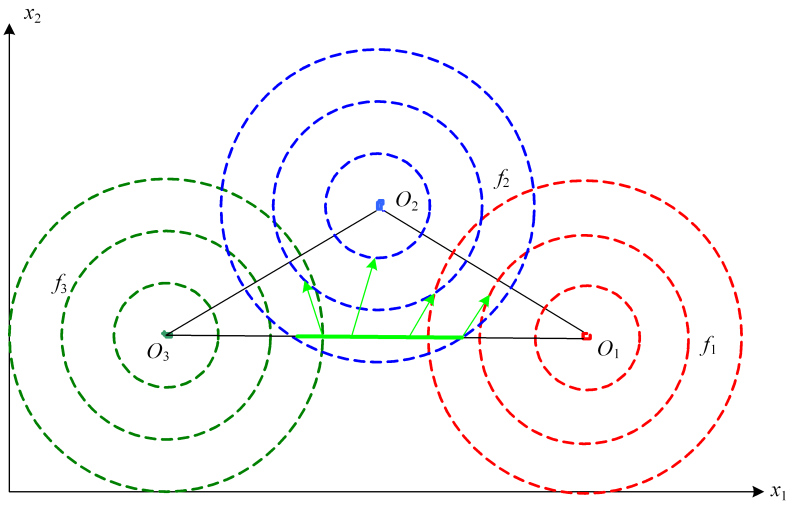


Figure 6.3: Three conflicting objectives with the same difficulties

solutions of the objective pairs are different. The objective pair $f_1 \bigcup f_3$ has a larger number of Pareto-optimal solutions than the other two pairs, as $O_1O_2 = O_2O_3 < O_1O_3$.

Actually, if the resources for evolution are more than enough, different ordering will not make big difference. However, the resources for intermediate phases of IMOO are limited to make it computationally efficient. Thus, if the Pareto set is too large, e.g. $f_1 \bigcup f_3$ is evolved first, it is very likely only part of it is found by MOO_1 and passed to MOO_2. Hence, the guidance in MOO_2 from the inherited solutions would be non-uniform, as shown by the light green arrows in Figure 6.3, which may result in incomplete or poorly distributed final Pareto set.

Therefore, the aims of our research are to evaluate the effect of objective ordering in incremental multi-objective problem solving, and to find out metrics for ranking the objective ordering so that the best objective order can be found.

6.1.2 Methodology

Due to the incremental nature and the limited resources used for the intermediate phases of IMOO, two factors associated with objective ordering are believed to have

impact on the final performance of IMOO. One is the conflict level between adjacent objectives and the other one is the difficulty in handling individual objectives.

However, these two factors are difficult to evaluate for the various forms of objective functions and there are no definitions about them in the literature. Thus, we have to use some indirect metrics for measurement of these two factors. In this thesis comparison-based methods to rank objective pairs and individual objectives are proposed for objective ordering.

The major idea is that the conflict level of each objective pair can be compared by the range of the Pareto front of each, and the difficulty in handling each individual objective can be compared by the performance of the optimization algorithm solving each. With the comparison results, the objective pairs can be ranked by their ranges (of the Pareto fronts) and the individual objectives can be ranked by their performance (of the optimization algorithm solving them). After ranking, we shall optimize as early as possible the objective pairs with smaller ranges and/or the objectives with better performance.

An ordering strategy is drawn comprising both metrics, after the investigation about the importance of each individual factor. Based on this strategy, the found best objective order should be consistent with the optimal order obtained by exhaustive searching.

With regard to the choice of metrics for evaluating the performance of IMOO, hyper-volume metric developed by Zitzler [89] is used with minor revision.

6.1.3 Hyper-volume Metrics for Performance Evaluation

As indicated by Zitzler [88], multi-objective optimization is quite different from single objective optimization in that there is more than one goal, including the

convergence to the Pareto-optimal front and the good (in most cases uniform) distribution of the solutions found. There has been an increasing interest in performance evaluation of multi-objective optimization and a lot of metrics have been proposed [117]. Since we aim at comparing the performance of IMOGA [118] with various objective orders, we need a metric that can measure the two goals mentioned above. The hyper-volume-based scaling-independent metrics developed by Zitzler [87] were chosen because they can measure the convergence to the pareto-optimal front, as well as the uniform distribution of solutions.

As described in [87], there are two volume-based metrics involved:

- S calculates the size of dominated objective space covered by a set of Pareto-optimal solutions.
- D calculates the difference in coverage by two sets of Pareto-optimal solutions. Let $A, B \subseteq X$ be two sets of Pareto-optimal solutions. The function D is defined by

$$D(A, B) = S(A, B) - S(B), \tag{6.1}$$

  which gives the size of the objective space dominated by A but not by B.

In the two-objective case, when a minimization problem is considered and the maximum values of f_1 and f_2 are equal to f_{1_max} and f_{2_max} respectively, the metrics mentioned above can be visualized as in Figure 6.4.

As shown in Figure 6.4, the shaded area represents $S(A)$, the objective space covered by A, and the area filled with diagonal represents $D(A, B)$, the objective space covered by A but not covered by B. Similarly, $S(B)$ and $D(B, A)$ also can be found in Figure 6.4.

Assume the Pareto-optimal front obtained by IMOGA is denoted as E, while the sampled set of the true Pareto-optimal front is denoted as T. To reveal the

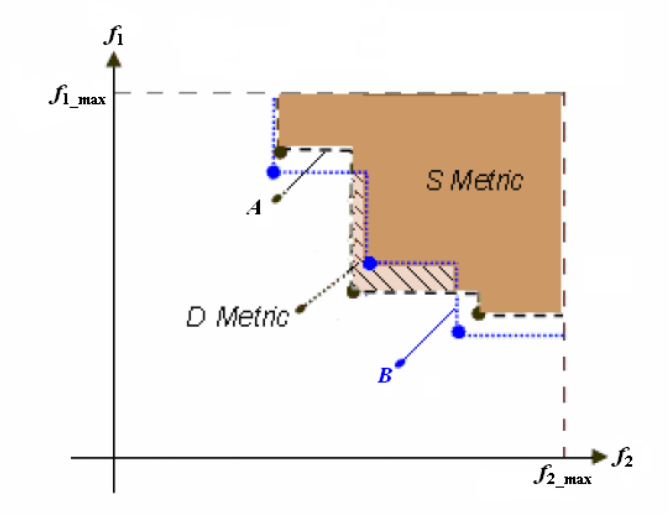


Figure 6.4: 2D Visualization of hyper-volume metrics

relationship of these two sets of Pareto-optimal solutions, we should consider both $D(E,T)$ and $D(T,E)$ according to the definition of D.

Ideally, $D(E,T)$ should always be zero as all the Pareto-optimal points should be contained in the true Pareto front. However, what should be noted is that some $D(E,T)$-error may be introduced by the sampling of the true Pareto-optimal front. As shown in Figure 6.5, $D(E,T) > 0$ because $P \in E$ but $P \notin T$. In this case, P is actually a point on the true Pareto-optimal front, as it cannot be dominated by any point in T. The reason why a non-ideal $D(E,T)$ happens is that an ideal true Pareto-optimal front contains all the non-dominated points in the objective space, while in practice a finite set of true Pareto-optimal points cannot completely cover the whole true Pareto-optimal front. Nevertheless, if the resolution of the sampled true Pareto-optimal front is high enough so that $|T| \gg |E|$, where $|\cdot|$ represents the number of points in a set, this kind of bias would be negligible. Therefore, the size of true Pareto front used to evaluate found Pareto fronts is set much larger than the desired solution number, namely the size of the found Pareto fronts, in the experiments of this chapter.

With regard to the Pareto-optimal front found by a multi-objective optimization algorithm, it is required that the true Pareto-optimal front should be covered

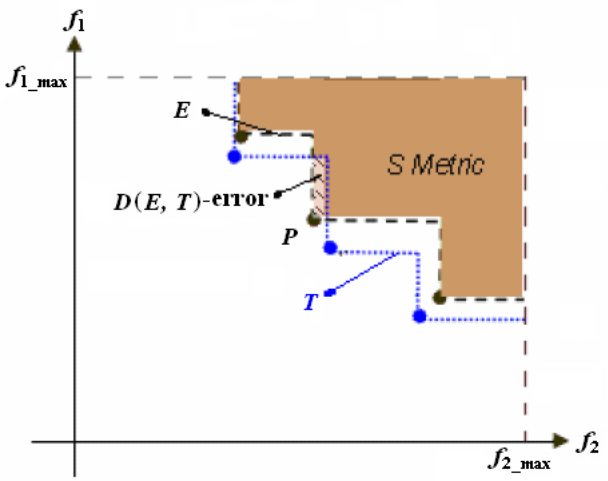


Figure 6.5: Hyper-volume error introduced by sampling

as much as possible. This means the $D(T, E)$ need to be minimized. Thus, we define a hyper-volume metric η to evaluate the performance of IMOO based on Zitzler's volume-based metrics,

$$\eta = \frac{\alpha}{V}, \tag{6.2}$$

where $\alpha = D(T, E)$ and $V = S(T)$. Using IMOO with various objective ordering to solve a certain multi-objective optimization problem, the less the value of η is, the better is the found Pareto-optimal front, thereby the better the objective order is. Thus, the values of η is used to compare the performance of IMOO with different objective order.

6.2 Rationale of Objective Ordering for IMOO

For the incremental model proposed in Chapter 5, the objective functions were introduced into the incremental evolution procedure following their original order. But we found that the objective order has significant impact on the final performance of IMOO. In this section, the effect of objective ordering will be illuminated and the principles for objective ordering will be concluded. Based on the discus-

sion, two objective ordering approaches will be proposed, which aim at finding the best objective order for IMOO to achieve its best performance.

6.2.1 Factors associated with Objective Ordering

For IMOO, the initial population after objective increment is generated by integrating the solutions obtained before objective increment and the solutions obtained by SOO on the newly added objective. We have analyzed in details the effect of objective increment on the Pareto fronts found before objective increment in Chapter 5. It has been shown that IMOO could benefit from the inheritance, as the solutions with good quality obtained in the earlier phases can help the search in the following phases. On the other hand, any inaccuracy incurred in the earlier phases may mislead the optimization in the following phases, resulting in premature or poor solution spread, and thereby offset the advantages of IMOO. The qualities of the solutions obtained by SOO and MOO fluctuate due to the following two factors respectively:

1. The difficulty of individual objectives

   It has been proved in [110] that most of the Pareto-optimal solutions will remain Pareto-optimal after objective increment. "Pareto-optimal" is defined for MOPs. With regard to the stage of SOO, in which only one single objective is involved, the concept of "Pareto-optimal" is equivalent to "global optimal". That is to say, the optimal solutions found in SOO in an earlier phase will remain Pareto-optimal during the following phases of IMOO. The more accurate these solutions are, the larger non-Pareto-optimal space they will dominate so that they could help eliminate non-Pareto-optimal solutions in the population more effectively. Thus, the objective whose global optimum is easier to find would better be evolved earlier.

2. The level of conflict among objectives

   For MOPs, the spread of solutions results from the conflict among objectives. Considering IMOO, the Pareto front is expected to be constructed part by part as the objectives will be handled one by one. This is realized by generating the initial population for a higher dimensional search by copying and integrating the solutions found in the one-dimensional lower search with the solutions found by SOO on the incremented objective. According to Theorem 5.1, the Pareto-optimality of the points found before objective increment will remain after objective increment. So, if the initial population of a certain phase is generated based on a more accurate Pareto set found in the previous phase, less search effort will be wasted on non-Pareto-optimal solutions. In other words, it would give more effective guidance on the search after objective increment and result in more accurate result which is the base of search for the next phase. That means this positive effect will ripple from one phase to the next till the ending phase. Therefore, objective pairs which are easier to get accurate results would better be optimized earlier, where high accuracy includes smaller distance to the true Pareto front, good coverage and spread of the solutions.

   There is no conflict problem in the initial phase because only one objective is involved. With regard to the intermediate phases, the difference resulted from different objective ordering can be seen from the following example. Given a 3-objective problem:

$$\begin{cases} f_1 = x^2 \\ f_2 = (x+1)^2 \qquad x \in [-2, 2], \\ f_3 = (x-1)^2 \end{cases} \tag{6.3}$$

   there are three phases, including the initial phase, the intermediate phase and the ending phase. To consider the conflict between the objective pair

which is evolved in the intermediate phase, there are three possible objective orders: 123, 132 and 231. Respectively, the objective pairs evolved in the intermediate phases are (12), (13) and (23) with the corresponding Pareto-optimal solutions $x \in [-1, 0]$, $x \in [0, 1]$ and $x \in [-1, 1]$. If we use ten bits to encode the solutions, the number of Pareto-optimal solutions obtained in the intermediate phases should be 256, 256 and 512 respectively. Given the same algorithm and the same amount of computation resources (including time), the intermediate phases having 256 solutions would tend to perform better than the one having 512 solutions, thereby pass more useful information to the ending phase and guide the search in the ending phase more effectively.

Since IMOO searches the objective space dimension by dimension, it will be computationally ineffective if abundant resources, like population size and number of generations, are wasted in the earlier phases. Thus, the computational effort, namely the number of function evaluations, for optimization in the initial and intermediate phases is limited to make IMOO efficient.

6.2.2 Principle of Objective Ordering

From the above analysis, it can be inferred that if we can rank the objectives according to their difficulty or rank the objective pairs according to their conflict, we may get good objective order for IMOO. Unfortunately, objective functions can be in various forms, continuous or discrete, differentiable or non-differentiable, convex or non-convex. Neither the conflict among the objectives nor the difficulties of them could be measured directly by any deterministic approach.

Nevertheless, it is believed that for a certain algorithm, given the same computation resources, the performance would get better as the task gets easier. Therefore, the difficulties of different individual objectives could be ranked according to

the qualities of the corresponding solutions obtained using the same optimization algorithm. Similarly, the conflicts in different objective sets could be ranked according to the characteristics of the corresponding Pareto-optimal solutions using the same multi-objective optimization algorithm. Theoretically, the single/multi-objective optimization algorithm used to rank the individual objectives/objective sets could be arbitrary. But we usually use algorithms in the same family with the algorithm to which the incremental model is applied for consistency. For instance, if the incremental model is applied to MOGA, resulting in IMOGA, we will use GA to get the rank of individual objectives and MOGA to get that of objective sets. Therefore, the metrics for ranking the objectives/objective sets can be summarized as follows:

1. SOO aims at finding solutions approaching the true global optimum of the individual objective as close as possible. So the quality of solutions, namely the difficulty of the corresponding objective, can be evaluated by their distance to the true optima.

2. MOO in any intermediate phase needs to find a Pareto-optimal front with a good spread. Since the spread of Pareto-optimal front results from the conflict among the objectives, the conflict in objective sets can be estimated and compared by the number of Pareto-optimal solutions obtained by a certain multi-objective optimization algorithm with certain computation resources. Since the number of solutions also depends on the resolution used, the same resolution should be used no matter what objective set is evaluated. The possible bias of this metric related to the resolution of solutions which will be further discussed in Section 6.5.

Based on these two metrics, two possible objective ordering approaches can be proposed. The detailed procedure will be described below.

6.3 Objective Ordering Approaches

For the ease of description, some definitions are given as follows:

1. **Relative Accuracy** (RA): For any objective in the objective set, the Euclidean distance between the solutions obtained by the associated SOO and the true optimum (obtained earlier), divided by the objective value of the true optimum, is called *relative accuracy*.

2. **Objective Pair**: A pair of any two objective functions is called an *objective pair*, which is denoted as (i, j), where $i, j = 1, 2, \ldots, n$, $i \neq j$ and n is the number of objective functions. So, the total number of objective pairs should be $\binom{n}{2} = \frac{n(n-1)}{2}$.

3. **Conflict Level** (CL): For a specific problem, the conflict level of each objective pair is assigned according to the number of Pareto-optimal points obtained by a certain algorithm. The more the Pareto-optimal points, the higher is the conflict level: CL of the objective pair having the smallest number of Pareto-optimal points is set to 1, CL of the objective pair having the second smallest number of Pareto-optimal points is set to 2, etc. $CL_{i,j}$ represents the conflict level of objective pair (i, j) and $CL_{i,j} = CL_{j,i}$.

4. **Cumulative Conflict Level** (CCL): Given a sequence of objectives and a new objective, *cumulative conflict level* is the sum of CLs between each objective in the original sequence and the new objective.

5. **Niche**: Each location in the sequence of objectives is called a *niche*. l_i represents the ith niche, where $i = 1, 2, \ldots, n$.

Based on the analysis given in Section 6.2, two objective ordering approaches are proposed for IMOO. Both of them consider the two ranking metrics RA and CL, but the priorities vary.

6.3.1 Difficulty based objective ordering approach (DOOA)

As mentioned, IMOO would prefer objectives with smaller RA being optimized earlier. DOOA is designed based on this observation as follows:

1. Calculate the relative accuracy of each objective, RA_i, $i = 1, 2, \ldots, n$.
2. Choose the objective order in ascending RA as the best order for IMOO. If ties appear, the objective that results in the smallest CL for the later stage will be chosen.

6.3.2 Conflict level based objective ordering approach (CLOOA)

As described in Section 6.1, IMOO would prefer objective pairs with smaller CL being optimized earlier. CLOOA is based on this observation. It is described as follows:

1. Rank the objective pairs according to the number of Pareto-optimal solutions found, and assign the rank of each objective pair as its conflict level, $CL_{i,j}$.
2. Choose an objective pair with the minimal CL to fill in l_1 and l_2.
3. Set $k = 0$.
4. $k = k + 1$. Scan the remaining objective functions to calculate the CCL, and choose the objective function corresponding to the smallest CCL to fill in l_{k+2}. According to the definition of CCL, when the pth objective function is considered, $CCL_{k,p} = \sum_{u=1}^{k+1} CL_{O_u,p}$, where O_u denotes the objective function which has been assigned to l_u. If ties appear, the objective with the smallest relative accuracy wins.
5. Repeat the procedure described in 4 until only one objective function is left and fill that objective function into l_n.

6.3.3 MOGA-based IMOO with objective ordering

With the extensive research of MOGAs, a lot of state-of-the-art MOGAs appeared such as SPEA, PAES and NSGA-II, and they have been widely used in various fields such as traffic control, industrial design and wireless communication. In contrast, MOPSO is relatively new in the area of multi-objective optimization, which is still under study. Thus, the MOGA-based IMOO, namely IMOGA, is used as a vehicle to investigate the objective ordering issue. Result of this research may be meaningful for all the IMOGAs, as well as shed light on other incremental algorithms.

To implement any of the two ordering approaches, values of both RA and CL are required. These can be computed based on the two ranking metrics before applying the complete procedure of IMOGA with objective ordering, which is shown by the flowchart in Figure 6.6.

The procedure of objective ordering induces some additional cost. If this cost is too high, it makes no sense to do the objective ordering. The complexity analysis is given as follows.

In the objective ordering procedure, the basic operations and their complexities in the worst case are (n is the number of objectives and m is the population size for 2-objective IMOGA):

- Run the 2-objective IMOGA for all the objective pairs: $O(\frac{n(n-1)}{2} \cdot 2m^2)$;
- Compute the RA values for all the objectives: $O(n)$;
- Assign the CL values for all the objective pairs: $O(\frac{n(n-1)}{2}) = O(n^2)$;
- Order the objectives according to their RAs and CLs: $O(n^3)$;

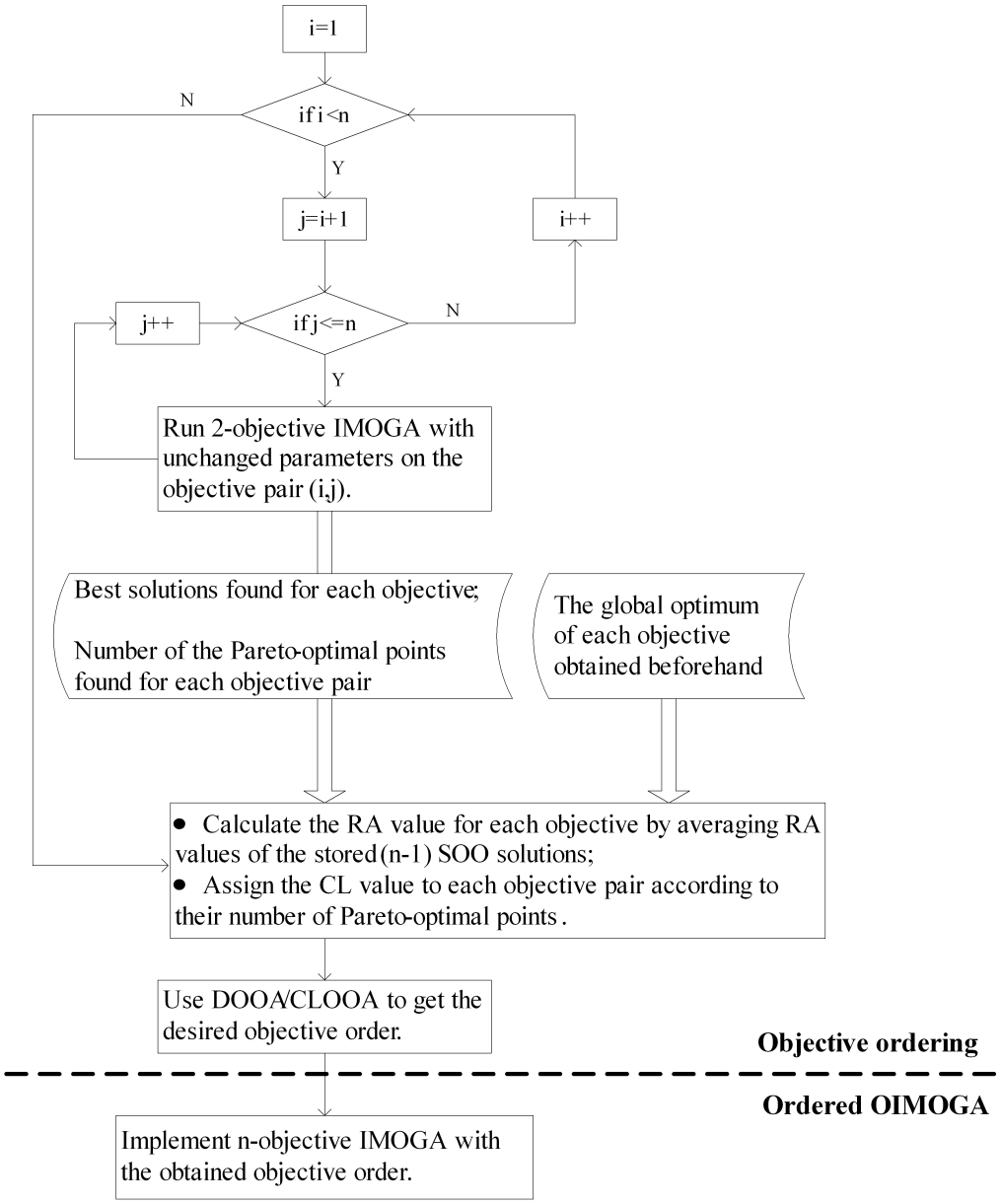


Figure 6.6: Flowchart of IMOGA with objective ordering

- Moreover, the global optimum of each single objective is required for the calculation of RA. Generally, the difficulty of solving MOPs comes from the conflict among objectives, rather than the individual objectives. So, mostly simple mathematical analysis can be made to get the global optima of each individual objective, which is what we have done for all the problems shown in this chapter. Even if such analysis is not feasible for some objectives, their global optima could be obtained by optimization algorithms, either decisive algorithms like linear programming or heuristic algorithms like GA.

In other words, the computation effort of obtaining the global optimum of each individual objective is a constant. Thus, the complexity of obtaining the global optimum of each objective is $O(n)$.

Normally, the population size used in MOGA is much larger than the objective number, $m \gg n$. Therefore, the total complexity of the objective ordering process is $O(n^2m^2)$.

On the other hand, the complexity of the n-objective IMOGA is $O(nM^2)$ (M is the population size for n-objective IMOGA) [110]. So the overall complexity of IMOGA with objective ordering is

$$\begin{cases} O(nM^2), & \text{if } \frac{M}{m} \geq \sqrt{n}, \\ O(n^2m^2), & \text{otherwise.} \end{cases} \tag{6.4}$$

Also, the population size is usually increased rapidly with the increase of objective number. For instance, in our experiments the population size used for 2-objective and 4-objective IMOGA is 100 and 1000, respectively. In other words, there is $M \gg m \gg n$. Thus, the condition $\frac{M}{m} \geq \sqrt{n}$ may be satisfied in most cases and the overall complexity of IMOGA with objective ordering shall be $O(nM^2)$. Since the computation complexities of IMOGA and other state-of-the-art MOGAs such as NSGA-II, SPEA and PAES are all shown to be $O(nM^2)$, the computation load of objective ordering is reasonable and acceptable.

As shown in Figure 6.6, we have to choose either DOOA or CLOOA for ordering. Although either in DOOA or in CLOOA the two ranking metrics are considered, apparently they are emphasized respectively. So we will test DOOA and CLOOA separately to find out which metric is crucial in deciding the objective ordering in the following section.

6.4 Experimental Results and Analysis

In this section, the ordering approaches proposed in Section 6.3 are validated by experiments. 6.4.1 Experimental Scheme Originally, IMOGA evolves an objective set in the original order. The aim of the experiments done in this chapter was to test whether the performance of IMOGA could be improved further by the proposed objective ordering approaches. Thus, a series of comparison experiments were designed. Firstly, the ordering approaches were tested with a 4-objective problem in which the difficulty of each objective is the same. Secondly, a 3-objective problem was tested, which reflects the situation of multiple objectives with the same conflict level among them. Lastly, two problems with different difficulties and conflict levels were tested. The general steps for comparison include:

1. The best objective order was obtained by each proposed ordering approach;
2. Implementing n-objective IMOGA with each of the $n!$ possible objective orders. The performance of IMOGA with each order was evaluated by the metric η as stated in Section 6.1.3. The objective order corresponding to the best performance was marked;
3. Comparing the best objective orders obtained in the above two steps respectively.

6.4.1 Experimental Scheme and evaluation metrics for 2-objective IMOGA

As described in Section 6.3, the ranking metrics RA and CL should be obtained by the 2-objective IMOGA. Given an objective pair, the 2-objective IMOGA is run 20 times with initial seeds from 1 to 20. The parameters of the 2-objective IMOGA were set as follows:

- For each problem, the algorithm is given 0.5s to evolve in each run.
- Each decision variable is encoded in 30 bits.
- Crossover: one-point crossover at the input variable boundary only, with a probability of 1.
- The mutation rate for each decision variable is $\frac{1}{d}$, and for each bit it is $\frac{1}{l}$ (d: the number of decision variables, l: the number of chromosome bits).
- Stopping criteria for single-objective evolution: the enhancement of the fittest individual is less than 0.1% in the last ten generations or the generation number is more than 1000.
- The initial population size for single-objective evolution is 100, and 25 best individuals are selected into the integration operation.
- The population size for multi-objective evolution is 1000.

All the parameters above follow the settings recorded in [118]. Chen has shown in [118] that the 2-objective IMOGA with these settings can achieve splendid performance in the sense that it can find a Pareto front very close to the true Pareto front with excellent spread and coverage. This guarantees the validity of the following evaluation metrics used for ranking:

1. N is the number of Pareto-optimal solutions obtained by the 2-IMOGA implemented on an objective pair. The larger the N, the higher conflict level will be assigned to this objective pair.
2. δ is the RA of each individual objective. The solutions obtained by the SOO steps of the 2-objective IMOGA are used to evaluate δ.

$$\delta = \frac{\sum_{j=1} k|y_j - Y|}{Y}, \tag{6.5}$$

where k is the number of solutions obtained by SOO associated with the objective, y_j is the objective value of the jth solution and Y is the true optimal value of this objective. The closer the solutions approaching the true optima, the less difficulty this objective possesses.

What should be noted is that:

1. Since the first two niches are equipotent, there is no ordering issue for 2-objective IMOGA.

2. For the fairness of comparison, the parameters of the 2-objective IMOGA should be fixed.

3. Since every possible objective pair should be evaluated, the number of SOOs associated with the same objective will be $n-1$. So, the repeated number of experiments for evaluating the difficulty of any individual objective is $20 \times (n-1)$.

4. It is assumed that the true optimum of each individual objective is known, which can be obtained either by numerical methods or well-received heuristics such as GAs and/or neural networks.

6.4.2 Experimental Results

With regard to the second step of the experimental scheme mentioned in Section 6.4.1, n-objective IMOGA will be applied to solve the test problems. The parameter setup for the n-objective IMOGA are kept unchanged from one objective ordering to another and roughly the same as the setup of the 2-objective IMOGA described in Section 6.4.2. The parameter setups that are different from one problem to another are listed in Table 6.1.

Table 6.1: Parameter setups for the n-objective IMOGAs used in each problem

Prob.	# of bits for encoding	Computation time (s)	Population size of SOOs	# of solutions inherited from SOOs	# of Gen.for inter. MOOs	Desired final solution #
1	8	3	20	5	5	256
2	10	10	100	15	5	1500
3	8	10	50	15	5	400
4	5	10	50	10	5	1000

Remarks:
'# of bits for encoding' is the number of bits used to encode each decision variable;
'Computation time' is the total time to which the n-objective IMOGA is limited;
'Population size of SOOs' is the number of chromosomes used in each SOO;
'# of solutions inherited from SOOs' is the number of solutions inherited from a SOE to form the initial population for the following MOO;
'# of Gen. for inter. MOOs' is the number of generations a MOO evolves its population;
'Desired final solution #' is the number of final Pareto-optimal solutions, also the population size of the last MOO.

Problem 1

To get rid of the influence of objective difficulty on performance, we set all the objectives with the same difficulties. In this way, how the conflict level affects the objective ordering can be seen clearly. A 4-objective problem is defined here.

$$\begin{aligned} f_1 &= (x_1 + 2)^2 + x_2^2, \\ f_2 &= (x_1 + 1)^2 + x_2^2, \\ f_3 &= (x_1 - 1)^2 + x_2^2, \\ f_4 &= (x_1 - 2)^2 + x_2^2, \end{aligned} \tag{6.6}$$

$$\text{with the constraints: } -2 \leq x_1, x_2 \leq 2.$$

For this problem, the conflict level between any objective pair is decided only by the first variable. The projections of the objectives along the first variable can be shown in Figure 6.7.

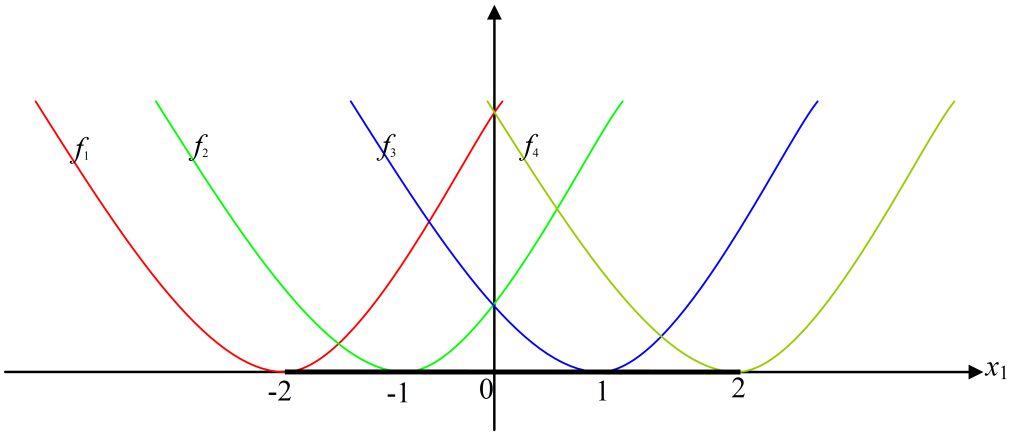


Figure 6.7: Projection of the objective functions in Problem 1

Table 6.2: True conflict level between objective pairs in Problem 1

Objective pair	12	13	14	23	24	34
Proportional range covered by corresponding Pareto solutions	$\frac{1}{4}$	$\frac{3}{4}$	1	$\frac{1}{2}$	$\frac{3}{4}$	$\frac{1}{4}$
CL	1	4	6	3	4	1

From Figure 6.7, it can observed that the Pareto set of Problem 1 covers the whole range of x_1. For different objective pairs, the coverage of the Pareto solutions and the corresponding conflict level are listed in Table 6.2.

As mentioned at the beginning of this section, the two-objective IMOGA was used to rank the objective pairs and assign conflict levels to them. The results are shown in Table 6.3.

Table 6.3: Assigned conflict level of all possible objective pairs in Problem 1

Objective Pair	12	13	14	23	24	34
N	256	768	1024	512	768	256
CL	1	4	6	3	4	1

Comparing the CL item in Table 6.2 and Table 6.3, it can be seen that the CL assignment obtained by the 2-objective IMOGA was consistent with the intuitive

analysis. Since the factor of objective difficulty had been eliminated, the two ordering approaches DOOA and CLOOA would get the same result and the ordering only depends on the CL. As remarked in Figure 5.2, the first two functions are equipotent and exchangeable. In this case, we only need to assign an objective pair to the first two niches without considering the order between those two objectives. Considering CLOOA, either (1,2) or (3,4) could be selected to fill in the first two niches l_1 and l_2, as they have the lowest CL. Say (1,2) is selected. Thereafter we have $CCL_{3,3} = 7$ vs. $CCL_{3,4} = 10$, which means selecting objective 3 for l_3 could benefit the third phase of IMOGA. Then the remaining objective 4 will fill in l_4. So **(12)34** is one of the best objective orderings obtained by the proposed ordering approach, where the brackets stand for exchangeability. Similarly, **(34)21** could be obtained as the other candidate.

To show the validity of the proposed approaches, the 4-objective IMOGA was used to solve the problem with different objective orders. The performance of this 4-objective IMOGA was measured by the hyper-volume metric η as described in Section 6.1.3. 256 true Pareto-optimal points were found beforehand by a brute-force method for evaluation of η. That is, each pair of two possible solutions in the feasible output space was compared to find those non-dominated solutions. Table 6.4 shows the average results over 20 runs with initial seeds from 1 to 20.

It should be noted that, since the first two niches are exchangeable, the slight difference of performance results due to the exchange of the first two objectives could be caused by the random initial setting. So, the performance of these two equipotent orders was averaged for comparison, denoted as $\overline{\eta}$. For the ease of comparison, the results in Table 6.4 were sorted in ascending order according to $\overline{\eta}$. As can be seen from Table 6.4, the order **(12)34** and the order **(34)21** give similar optimal performance. So, both the orders are regarded as optimal orders.

Table 6.4: Performance comparison of 4-objective IMOGA with different objective orders for Problem 1

Order of objectives	η	$\overline{\eta}$
1234	0.076958	0.076979
2134	0.077000	
3421	0.077001	0.077074
4321	0.077147	
3214	0.119999	0.124236
2314	0.128472	
2341	0.127594	0.124355
3241	0.121115	
3124	0.226651	0.226561
1324	0.226471	
4231	0.226374	0.227425
2431	0.228476	
1423	0.382944	0.391459
4123	0.399974	
4132	0.398629	0.399942
1432	0.401254	
1243	0.572139	0.575882
2143	0.579624	
3412	0.697808	0.700969
4312	0.704129	
1342	0.860990	0.853981
3142	0.846971	
2413	0.887645	0.854828
4213	0.822011	

From this experiment, it can be concluded that the optimal objective orders found by the proposed ordering approaches are consistent with the optimal objective orders found by exhaustive search using 4-objective IMOGA.

Problem 2

It is a 3-objective objective ordering problem as shown below:

$$\begin{aligned} f_1 &= x_1x_2 + x_2x_3 + x_3x_4, \\ f_2 &= 1 + (x_1 - x_2)^2 + (x_3 - x_4)^2, \\ f_3 &= 1 - \exp(\frac{x_1 + 2x_2 + 3x_3 + 4x_4}{10}), \end{aligned} \tag{6.7}$$

with the constraints: $1 \leq x_1, x_2, x_3, x_4 \leq 10$.

The true optimal value of each objective in Problem 2 is: $f_{1\_min} = 3$, $f_{2\_min} = 1$ and $f_{3\_min} = -22025.5$, respectively.

Firstly, three 2-objective problems corresponding to the three possible objective pairs were solved by the 2-objective IMOGA. And they were ranked according to their number of Pareto-optimal solutions, resulting in the conflict levels. The results are shown in Table 6.5.

Table 6.5: Conflict level of all possible objective pairs in Problem 2

Objective Pair	12	13	23
N	1	550	1
CL	1	2	1

It can be seen from Table 6.5 that there are two objective pairs with the same CL. In this case, it can be seen clearly how the objective difficulty impact on the

Table 6.6: Relative accuracy of each SOO in Problem 2

Objective associated with SOO	f_1	f_2	f_3
δ	0.021713	0.00024	0.010004

objective ordering. On the other hand, the relative accuracy of each SOO, namely the difficulty of each objective, is evaluated and the results are shown in Table 6.6.

Secondly, we can find from Table 6.6 that the best objective ordering based on DOOA should be **(23)1**. The steps of CLOOA are described as follows:

1. We chose the objective pair with the smallest CL. There was a tie between (1,2) and (2,3). So, the objectives difficulty should be considered.
2. Since $\delta_3 < \delta_1$, (2,3) was chosen for the first two niches.
3. The third niche was filled in with the remaining objective 1.

Thus, **(23)1** is the optimal objective order.

Lastly, to show the validity of the proposed approaches, a 3-objective IMOGA was used to solve the problem with different objective orders. 9457 true Pareto-optimal points were found beforehand by a brute-force method for the evaluation of η. The results are shown in Table 6.7, which shows the average over 20 runs with initial seeds from 1 to 20. As can be seen from Table 6.7, the optimal order is **(23)1**.

From this experiment, it can be concluded that the optimal objective orders found by both DOOA and CLOOA were consistent with the true optimal objective order found by exhaustive searching using 3-objective IMOGA.

Table 6.7: Performance comparison of 3-objective IMOGA with different objective ordering for Problem 2

Order of objectives	123	213	132	312	231	321
η	0.019299	0.019055	0.022095	0.0228232	0.017338	0.018892
$\overline{\eta}$	0.019177		0.022459		0.018115	

Problem 3

It is a 3-objective objective ordering problem as shown below:

$$\begin{aligned} f_1 &= 2 - \exp(-\sum_{i=1}^{3}(x_i - \frac{1}{\sqrt{3}})^2), \\ f_2 &= 2 - \exp(-\sum_{i=1}^{3}(x_i + \frac{1}{\sqrt{3}})^2), \\ f_3 &= x_1 + x_2 + x_3, \end{aligned} \tag{6.8}$$

with the constraints: $-4 \leq x_1, x_2, x_3 \leq 4$.

The true optimal value of each objective in Problem 3 is: $f_{1\_min} = 1$, $f_{2\_min} = 1$ and $f_{3\_min} = -12$, respectively.

The conflict level of each objective pair and the accuracy of each individual objective are shown in Table 6.8 and Table 6.9, respectively.

Table 6.8: Conflict level of all possible objective pairs in Problem 3

Objective Pair	12	13	23
N	98	149	168
CL	1	2	3

From Table 6.9 we can find that the best objective ordering based on DOOA should be **(23)1**. To apply CLOOA, we chose the objective pair with the smallest

Table 6.9: Relative accuracy of each SOO in Problem 3

Objective associated with SOO	f_1	f_2	f_3
δ	0.001899	0.001740	0.001716

CL based on Table 6.8 to fill in the first two niches, which is (1,2). So, the third niche is filled in with the remaining objective 3. Thus, CLOOA gave **(12)3** as the optimal objective order for handling this problem by IMOGA.

To validate the proposed approaches, a 3-objective IMOGA is used to solve the problem with different objective orders. A total of 378 true Pareto-optimal points were found beforehand by a brute-force method for evaluation of η. The results are shown in Table 6.10, which shows the average over 20 runs with initial seeds from 1 to 20. As can be seen from this table, the optimal order is **(12)3**.

Table 6.10: Performance comparison of 3-objective IMOGA with different objective ordering for Problem 3

Order of objectives	123	213	132	312	231	321
η	0.009875	0.009561	0.012139	0.012067	0.012497	0.013045
$\overline{\eta}$	0.009718		0.012103		0.012771	

Comparing the experimental results, it can be concluded that the optimal objective order found by CLOOA was consistent with the true optimal objective order found by exhaustive searching using 3-objective IMOGA, while the optimal objective order found by DOOA was not.

Problem 4

It is a 4-objective objective ordering problem as shown below:

$$\begin{aligned} f_1 &= (x_1 - 2)^2 + 4x_2^2, \\ f_2 &= x_1^2 + (x_2 - 3)(x_3 - 3), \\ f_3 &= x_1 x_2 x_3, \\ f_4 &= \frac{1}{x_2^{1.5} x_3^{2.5} x_4}, \end{aligned} \tag{6.9}$$

$$\text{with the constraints: } 1 \leq x_1, x_2, x_3, x_4 \leq 10.$$

The true optimal value of each objective in Problem 4 is:$f_{1\_min} = 4$, $f_{2\_min} = -13$, $f_{3\_min} = 1$ and $f_{4\_min} = 10^{-5}$, respectively. The conflict level of each objective pair and the accuracy of each individual objective are shown in Table 6.11 and Table 6.12, respectively.

Table 6.11: Conflict level of all possible objective pairs in Problem 4

Objective Pair	12	13	14	23	24	34
N	93	1	363	361	277	408
CL	2	1	5	4	3	6

Table 6.12: Relative accuracy of each SOO in Problem 4

Objective associated with SOO	f_1	f_2	f_3	f_4
δ	0.004209	0.002456	0.0014516	0

According to the rules of DOOA, the best objective order should be **(43)21**. On the other hand, the ordering steps based on CLOOA, are shown in Figure 6.8.

To validate the proposed ordering approaches, a 4-objective IMOGA was used to solve the problem with different objective orders. A total of 6224 true Pareto-optimal points were found beforehand by the brute-force method for the evaluation

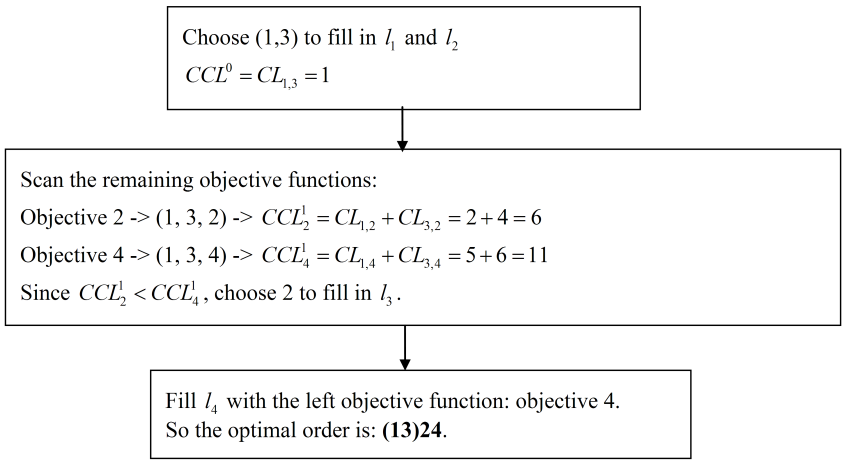


Figure 6.8: Procedure of CLOOA

of η. Table 6.13 shows the results, which are the average over 20 runs with initial seeds from 1 to 20. For the ease of comparison, the results in Table 6.13 were sorted in ascending order according to $\overline{\eta}$. It can be seen that the optimal order is **(13)24**, which has the lowest value of $\overline{\eta}$.

In this problem, it can be seen that the optimal objective orders found by DOOA and CLOOA are different, and only the one obtained by CLOOA is consistent with the optimal objective order found by exhaustive searching using 4-objective IMOGA.

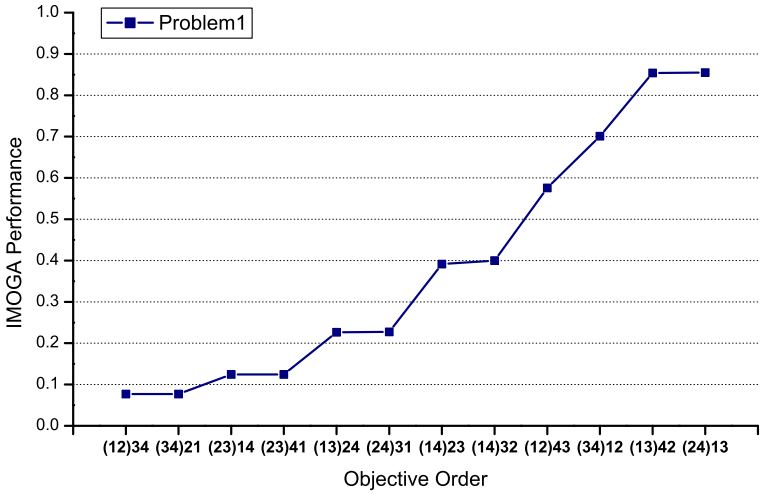
6.4.3 Analysis of the experimental results

From the experimental results above, it can be observed that:

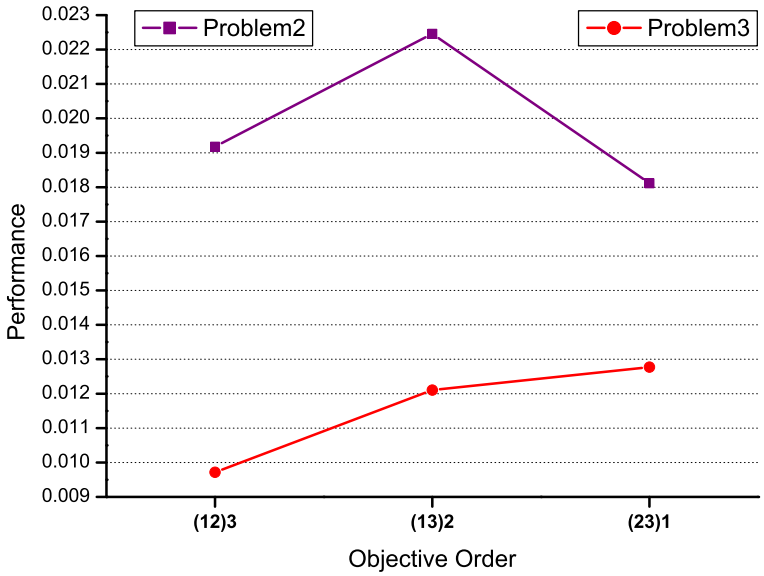
1. As shown in Figure 6.9, the objective ordering for IMOGA does have impact on the final performance of the IMOGA, and the variance of performance would become clearer as the number of objectives increases. So it is important to find the optimal objective order to get the best performance of IMOGA, especially when the number of objectives is large.

Table 6.13: Performance comparison of 4-objective IMOGA with different objective orders for Problem 4

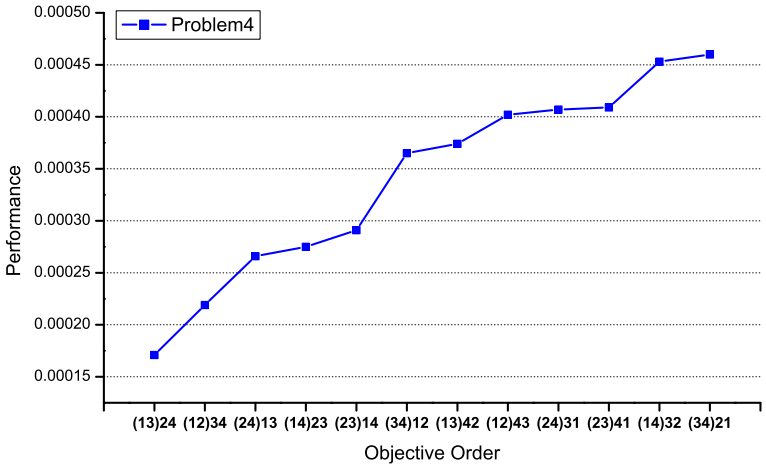
Order of objectives	η	$\overline{\eta}$
3124	0.000153	0.000171
1324	0.000189	
1234	0.000188	0.000219
2134	0.000250	
4213	0.000270	0.000266
2413	0.000263	
4123	0.000270	0.000275
1423	0.000280	
2314	0.000291	0.000291
3214	0.000291	
3412	0.000387	0.000365
4312	0.000343	
1342	0.000328	0.000374
3142	0.000420	
2143	0.000386	0.000402
1243	0.000417	
2431	0.000395	0.000407
4231	0.000419	
2341	0.000406	0.000409
3241	0.000411	
1432	0.000509	0.000453
4132	0.000398	
4321	0.000461	0.000460
3421	0.000458	



(a) Results in Problem 1



(b) Results in Problem 2 and 3



(c) Results in Problem 4

Figure 6.9: Performance of IMOGA measured by hyper-volume metrics under different objective orders from Problem 1 to Problem 4

2. The metric CL plays a decisive role in determining the ordering for IMOGA. CLOOA can find the optimal objective order from Problem 1 to Problem 4 without exception.

3. The metric RA could be used as a secondary factor in considering the ordering for IMOGA. For the tested problems, DOOA only finds the optimal objective order for Problem 2. However, we cannot get rid of this metric, as it would help determine the ordering in the case of CL tie which can be seen in Problem 2.

4. The time required in the ordering process comprises of getting the optimum of each objective, running 2-objective IMOGAs and calculating values of the two ranking metrics. Considering GA is used to search the optimum of the single objectives in the test problems, it will not take more than 0.2s for any one. Each 2-objective IMOGA was given 0.5s to run. The time required to evaluate the two ranking metrics was extremely short, which can be ignored. Therefore, the computation loads of objective ordering for 3-objective problem and 4-objective problem are about $0.2 \times 3 + 0.5 \times 3 = 2.1s$ and $0.2 \times 4 + 0.5 \times 6 = 3.8s$. Compared to the computation time taken for the n-objective IMOGA, these loads are acceptable, which is consistent with the complexity analysis presented in Section 6.3.3.

Therefore, it can be concluded that IMOGA can achieve its potentially best performance in all the test cases with objective ordering at an acceptable cost. CLOOA is an approach to objective ordering that can find the optimal objective order for IMOGA.

6.5 Discussion

The reason why the difficulty of individual objective has less influence in objective ordering may be that the bias incurred by the error of SOOs has chances to be rectified. Assume an objective with great difficulty is evolved in an early phase. Even if the SOO associated with it is likely to be trapped and the local optima will be inherited into the subsequent MOOs, once the global optimum of the objective is found in the MOOs all the biased Pareto-optimal solution resulting from that local optima will be dominated and discarded. In contrast, the poor spread of Pareto-optimal solutions incurred by the objective set with high conflict level could not be improved because those solutions are non-dominated to each other and will not be discarded. Thus, the conflict level plays a decisive role in determining the ordering for IMOO.

Besides, there are two issues that should be noted. One is the equipotence of the first two niches. It can be seen from the incremental model given in Chapter 5 that the positions of the first two niches are equipotent and exchangeable. But why the IMOO performance varies slightly by exchanging the first two objectives? In fact, this is because we feed the same initial seeds into IMOO with different objective ordering for fairness of comparison. In this case, the series of random numbers generated in each IMOO is the same. So, exchanging the first two objectives is equivalent to exchanging the initial populations for the first two SOO, which results in slightly different final performance.

The other issue is the distribution and resolution of solutions. To use the number of solutions N as the metric for evaluating the conflict level, the solutions need to be distributed uniformly and the resolution of solutions should be kept unchanged. This means that the number of bits used to encode the chromosomes should be fixed for a problem. Otherwise, the metric N may be biased because using either a fixed resolution to sample surfaces with uneven distributions or different

resolutions to sample uniformly distributed surfaces will result in different number of solutions.

Chapter 7

Conclusions and Future Work

This chapter briefly summarizes the findings and contributions of this thesis, and highlight some interesting and meaningful directions for future research.

7.1 Contributions

The primary goal of the present thesis was to build models of incremental optimization both in the input space and in the output space for CIAs. To make the proposed models concrete, we apply them to PSO that acts as a vehicle for validation. The obtained PSO-based incremental algorithms are compared with non-incremental algorithms in order to investigate their advantages and limitations. The achievements and contributions of this thesis are given in detail as follows.

7.1.1 Incremental optimization in the input space

- A solid mathematical foundation was provided for the incremental global optimization, based on which an incremental model was built in the input

space. This model allows CIAs to optimize from low dimensional spaces and then move to higher dimensional space incrementally, which could benefit the CIAs in terms of increasing their probability of global convergence.

- Novel PSO-based incremental optimization algorithms, IPSO and PIES, were designed and implemented based on the incremental model built in the input space. Experiments have shown that the PSO-based incremental algorithms are superior to the non-incremental algorithms on the benchmark function tested. This result suggests the use of the incremental model to improve the performance of canonical PSO. Experimental results also showed that PIES (the incremental algorithm generated by a hybrid implementation of the incremental model) obtained better performance than IPSO (the pure PSO-based incremental algorithm). This finding sheds light on the study of incremental algorithms that we may combine different CIAs under the incremental model to generate a powerful hybrid incremental algorithm.

- A parallel implementation of the IPSO was realized by using a BBS mechanism. In this parallel IPSO (PIPSO), the information gained from searching in the spaces with reduced dimensionality could be shared with a higher efficiency than in the original IPSO. According to the fact that the rate of convergence of the PIPSO was much higher compared to IPSO, we can infer that the parallel incremental implementation may improve the efficiency of the incremental model.

7.1.2 Incremental optimization in the output space

- For incrementally solving MOPs, the relationship between the Pareto fronts before and after objective increment was analyzed and the rationale behind

the incremental optimization in output space, i.e. incremental multi-objective optimization, was stated. Based on this rationale, an incremental model in the output space was built. This model was supposed to help the multi-objective CIAs obtain better Pareto-optimal fronts.

- A novel PSO-based incremental multi-objective optimization, IMOPSO, was designed and implemented based on the incremental model built in the output space. Experiments on IMOPSO showed that they are superior to the non-incremental algorithms, even in a real-world MOP with five objectives. This finding is of considerable importance for PSO-based multi-objective optimization, as most of the published work on MOPSO algorithms considers MOPs with two objectives.

- The issue of objective ordering was investigated, which has influence on the incremental multi-objective optimization. Considering the influence, two factors were detected. Based on these factors, two objective ordering approaches (DOOA and CLOOA) were proposed with different focus. These approaches aim at finding the optimal objective order so that the incremental multi-objective optimization is able to achieve its potential best performance. According to the results of validation experiments, CLOOA was found successful in achieving the target. In addition, the complexity of CLOOA was analyzed. With the result, we can predict that it is comparable with that of the incremental multi-objective algorithms. Therefore, it may be worthy to use CLOOA to obtain a good objective ordering before conducting the incremental algorithms.

The benchmark problems used in this thesis have various characteristics, which represent the difficulties encountered in real-world optimization problems. Thus,

the successful application of incremental algorithms on the benchmark problems implies that the proposed incremental models would be helpful for solving real-world applications. In addition, the implementation of the incremental models is problem-independent. That is to say, they can be used regardless of the form of objective function. The case study on a 5-objective MOP in Chapter 5 is a good instance. There are even no explicit objective functions. The objective values are obtained in indirect way. The incremental algorithms still can work and obtain outstanding performance.

7.2 Future Work

This thesis opens up a number of interesting directions for further investigation. We describe some of them as follows:

- We have studied the ordering issue for the incremental multi-objective optimization and obtained some guidelines. Similarly, the ordering issue also exists for the incremental model in the input space, regarding the incrementally presented variables. With different orders of the decision variables, the performance of an incremental algorithm may vary, therefore an ordering approach may be required to configure the order before conducting the incremental algorithm.

- The PIES has been proved an efficient hybrid implementation of the incremental model, which combines two CIAs, the PSO and (1+1)-ES. An in-depth analysis on this success may reveal how to choose CIAs for implementing different components of the incremental model. With such guidelines, more and more powerful incremental algorithms could be generated.

- It has been shown that the performance of IPSO can be largely improved with the parallel implementation. However, IPSO's ability in handling functions with highly correlated variables may be weakened by applying the parallel model, since the MVOs that adjust simultaneously more than one variable are discarded. So, it may be necessary to insert a layer of MVOs between the SVOs and BBS of the parallel model. However, adding a layer means more complicated procedure with increased number of modules. With the precondition that the total number of function evaluations is a fixed number to limit the computational cost, how to allocate the number of function evaluations would be an important issue to be considered.

- In the present study, the population size and number of function evaluations are fixed at values obtaining from trials and experience. In the future, studies on the dynamic configuration of the resources may be performed, so that the resource allocation could become more reasonable, efficient and reliable.

- So far, we have focused on the "divide-and-conquer" scheme all along. More specifically, we always started from solving decomposed (or projected) problems, and then used certain form of aggregation (or integration) to increase the dimensionality of search. We can consider inverting the process, so that dealing with the original problem at the beginning, followed by refinement in the search spaces with reduced dimensionality. By this approach, some suboptimal solutions will be found first. In other words, the good regions in search space are detected. Then, exploration will be performed around them in the subspaces. This may be especially useful for problems with correlated variables, as all the variables are optimized together at the beginning.

Bibliography

[1] E. Hansen, *Global Optimization Using Interval Analysis*. Dekker, New York 1992.

[2] V. Cerny, "A thermodynamical approach to the travelling salesman problem: an efficient simulation algorithm," *Journal of Optimization Theory and Applications*, vol. 45, pp. 41–51, 1985.

[3] S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[4] T. Bäck, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*, Oxford University Press, 1996.

[5] T. Bäck, D. Fogel, Z. Michalewicz, *Handbook of Evolutionary Computation*, Oxford University Press, 1997.

[6] A. E. Eiben, J. E. Smith, *Introduction to Evolutionary Computing*, Springer, 2003.

[7] E. Bonabeau, M. Dorigo and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*, Oxford University Press, 1999.

[8] J. Kennedy, R. C. Eberhart and Y. H. Shi, *Swarm Intelligence*, San Francisco; London, Morgan Kaufman Publishers, 2001.

[9] H. Tamaki, H. Kita and S. Kobayashi, "Multi-objective optimization by genetic algorithms: A review," *Proceedings of the IEEE Conference on Evolutionary Computation (ICEC'96)*, pp. 517-522, Piscataway, NJ, May 20-22, 1996.

[10] A. Carlos and C. Coello, "Handling Multiple Objectives With Particle Swarm Optimization", *IEEE Transactions on Evolutionary Computation*, Vol. 8, No. 3, pp. 256-279, 2004.

[11] A. Charnes and W. W. Cooper, *Management Models and Industrial Applications of Linear Programming*, vol. 1, Wiley, New York.

[12] M. J. Matarić, "Designing and Understanding Adaptive Group Behavior", *Adaptive Behavior*, vol. 4, pp. 51-80, 1995.

[13] E. M. L. Beale. *Introduction to Optimization*. John Eiley & Sons Ltd., 1988.

[14] E. Polak, *Optimization: Algorithms and Consistent Approximations*, Springer-Verlag, New York, USA, 1997.

[15] M. F. Möller, "A scaled conjugate gradient algorithm for fast supervised learning", In *Neural Networks*, vol. 6, pp. 525-533, 1993.

[16] H. Tuy, *Convex Aalysis and Gobal Otimization*. Kluwer Academic Publishers, Netherlands, 1998.

[17] J. C. Spall. *Introduction to Stochastic Search and Optimization*. Hoboken, N.J.: Wiley-Interscience, 2003.

[18] B. D. Hughes, *Random walks and random environments*. Oxford University Press, 1996.

[19] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, Prentice-Hall, 1995.

[20] L. C. W. Dixon and G. P. Szegø, editors, *Towards Global Optimization*, North-Holland, Amsterdam, 1975.

[21] L. C. W. Dixon and G. P. Szegø, editors, *Towards Global Optimization*, North-Holland, Amsterdam, 1978.

[22] M. Ehrgott, *Multiple Criteria Optimization: Classification and Methodology.* Shaker Verlag, 1997.

[23] K. Gurney , *An Introduction to Neural Networks*, London: Routledge, 1997.

[24] J. Robinson, S. Sinton and Y. R. Samii, "Particle swarm, genetic algorithm, and their hybrids: optimization of a profiled corrugated horn antenna", *IEEE Antennas and Propagation Society International Symposium and URSI National Radio Science Meeting*, San Antonio, TX, 2002.

[25] X. F. Xie, W. J. Zhang and Z. L. Yang, "A dissipative particle swarm optimization, Congress on Evolutionary Computation", In *Proceedings of the Congress on Evolutionary Computation*, pp. 1456-1461, 2002.

[26] T. I. Cristian, "The particle swarm optimization algorithm: convergence analysis and parameter selection", *Information Processing Letters*, vol. 85, no. 6, pp. 317-325, 2003.

[27] S. U. Guan and P. Li, "Incremental Learning in Terms of Output Attributes", *Journal of Intelligent Systems*, vol. 13, no. 2, pp. 95-122, 2004.

[28] T. Bäck, *Evolutionary Algorithms in Theory and Practice*, Oxford University Press, 1996.

[29] A. E. Eiben, T. Back, "Empirical Investigation of Multiparent Recombination Operators in Evolution Strategies", *Journal of Evolutionary Computation*, vol. 5, no. 3, pp. 345-365. 1997.

[30] D. Whitley, "An Overview of Evolutionary Algorithms: Practical Issues and Common Pitfalls", *Journal of Information and Software Technology*, vol. 43, pp. 817-831. 2001.

[31] J. Kennedy, "The particle swarm: social adaptation of knowledge", In *Proceedings of the IEEE International Conference on Evolutionary Computation*, pp. 303-308, 1997.

[32] J. Kennedy, "Bare bones particle swarms", *Intelligence Symposium*, pp. 80-87, 2003.

[33] C. L. Hwang and K. Yoon, *Multiple Attribute Decision Making, Methods and Application, a State-of-Art Survey*, Springer-Verlag, New York, 1981.

[34] M. Zeleny, *Multiple Criteria Decision Making*, McGraw-Hill, New York, 1982.

[35] K. Deb, A. Pratap, S. Agarwal and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182C197, Apr. 2002.

[36] N. Srinivas and K. Deb, "Multiobjective optimization using non-dominated sorting in genetic algorithms," *Evolutionary Computation*, vol. 2, no. 3, pp. 221-248, 1994.

[37] D. Dumitrescu, B. Lazzerini ,L. C. Jain and A.nDumitrescu, *Evolutionary Computation*. CRC Press LLC, Florida, 2000.

[38] J. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, 1975.

[39] D. Goldberg, *Genetic Algorithms in Search*, Optimization and Machine Learning. Addison-Wesley, Reading, MA, 1989.

[40] H. P. Schwefel, *Numerical Optimization of Computer Models*, Wiley. 1981.

[41] H. P. Schwefel, *Evolution and Optimum Seeking*. Wiley, 1985.

[42] D. B. Fogel, "Evolutionary programming: an introduction and some current directions". *Statistics and Computing*, vol. 4, pp. 113-129, 1994.

[43] D. B. Fogel, *Evolutionary Computation*, IEEE Press, New York, 1995.

[44] J. Kennedy and R. C. Eberhart, "Particle swarm optimization", In *Proceedings of the IEEE International Conference on Neural Networks* (Perth, Australia), IEEE Service Center, Piscataway, NJ, pp. 1942-1948, 1995.

[45] M. A. Potter, *The Design and Analysis of a Computational Model of Cooperative Coevolution*, PhD thesis, George Mason University, Fairfax, Virginia, USA, 1997.

[46] L. J. Eshelman and J. D. Schaffer, "Real-coded genetic algorithms and interval schemata", In *Foundations of Genetic Algorithms II*, pp. 187-202, Morgan Kaufmann, San Mateo, CA, USA, 1993.

[47] K. Deb and R. Agrawal, "Simulated Binary Crossover for Continuous Search Space", *Complex Systems*, vol. 9, pp. 115-148, 1995.

[48] H. P. Schwefel, *Kybernetische Evolution als Strategie der experimentellen Forschung in der Strømungstechnik*, Techniche Universität, Serlin, 1965.

[49] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory", In *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pp. 39-43, Nagoya, Japan, 1995.

[50] H. P. Schwefel, "Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrregie", In *Interdisciplinary Systems Research*, Birkhäuser, Basel, 1977.

[51] I. Rechenberg, "Cybernetic Solution Path of an Experimental Problem", *Library translation No 1122*, Royal Aircraft Establishment, Farnborough, UK, 1965.

[52] T. Bäck and H. P. Schwefel, "An overview of evolutionary algorithms for parameter optimization", *Evolutionary Computation*, vol. 1, no. 2, pp. 101-125, 1993.

[53] I. Rechenberg, *Evolutions strategie:Optimierung technischer Systeme nach Prinzipien der Biologischen Evolution*, Frommann-Holzboog, Stuttgart, 1973.

[54] van den Bergh, *An Analysis of Particle Swarm Optimizers*, Doctor dissertation, University of Pretoria, Pretoria, 2001.

[55] van den Bergh and F. Engelbrecht, "A Study of Particle Swarm Optimization Particle Trajectories", *Information Sciences*, vol. 176, no. 8, pp. 937-971, Apr. 2006.

[56] van den Bergh and F. Engelbrecht, "A Cooperative Approach to Particle Swarm Optimisation", *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 225-239, Jun. 2004.

[57] S. H. Clearwater, T, Hogg and B. A. Huberman, "Cooperative Problem Solving", In *Computation: The Micro and Macro View*, pp. 33-70, Singapore: World Scientific, 1992.

[58] D. Corne, M. Dorigo and F. Glover, editors, *New Ideas in Optimization*, chapter 14, pp. 217-279, McGraw Hill, 1999.

[59] P. Grosso, *Computer Simulations of Genetic Adaption: Parallel Subcomponent Interaction in a Multilocus Model*, PhD thesis, University of Michigan, 1985.

[60] M. A. Potter and K. A. de Jong, "A Cooperative Coevolutionary Approach to Function Optimization", In *The Third Parallel Problem Solving from Nature*, pp. 249-257, Jerusalem, Israel, 1994.

[61] J. Moore and R. Chapman, "Application of Particle Swarm to Multiobjective Optimization", *Technical report*, Dept. Comput. Sci. Software Eng., Auburn Univ., 1999.

[62] T. Ray and K. M. Liew, "A swarm metaphor for multiobjective design optimization", *Eng. Opt.*, vol. 34, no. 2, pp. 141C153, Mar. 2002.

[63] J. E. Fieldsend and S. Singh, "A multi-objective algorithm based upon particle swarm optimization, an efficient data structure and turbulence", In *Proceedings of 2002 U.K. Workshop on Computational Intelligence*, pp. 37C44, Birmingham, U.K., Sept. 2002.

[64] S. Mostaghim and J. Teich, "Strategies for finding good local guides in Multi-Objective Particle Swarm Optimization (MOPSO)", In *Proceedings of 2003 IEEE Swarm Intelligence Symposium*, pp. 26C33, Indianapolis, IN, Apr. 2003.

[65] C. A. Coello Coello, D. A. Van Veldhuizen and G. B. Lamont, *Evolutionary Algorithms for Solving Multi-Objective Problems*, Norwell, Kluwer, MA, 2002.

[66] X. Li et al., "A nondominated sorting particle swarm optimizer for multiobjective optimization", In *Proceedings of Genetic and Evolutionary Computation (GECCO 2003)*, part 1, pp. 37C48, Berlin, Germany, July, 2003.

[67] J. D. Knowles and D.W. Corne, "Approximating the nondominated front using the Pareto archived evolution strategy", *Evolutionary Computation*, vol. 8, pp. 149-172, 2000.

[68] X. H. Hu and R. Eberhart, "Multiobjective optimization using dynamic neighborhood particle swarm optimization", In *Proceedings of the Conference*

on Evolutionary Computation (CEC2002), vol. 2, pp. 1677-1681, Honolulu, HI, May 2002.

[69] X. H. Hu, R. C. Eberhart, and Y. Shi, "Particle swarm with extended memory for multiobjective optimization", In *Proceedings of 2003 IEEE Swarm Intelligence Symposium*, pp. 193C197, Indianapolis, IN, Apr. 2003.

[70] P. J. Angeline, "Evolutionary optimization versus particle swarm optimization: philosophy and performance difference", In *Proceedings of the Annual Conference on Evolutionary Programming*, pp. 601-610. 1998.

[71] T. Bäck and D. Fogel, *Handbook of Evolutionary Computation*, 1997.

[72] Y. H. Shi and R. C. Eberhart, "Fuzzy Adaptive Particle Swarm Optimization", In *Proceedings of IEEE Int. Conf. on Evolutionary Computation*, pp. 101-106, 2001.

[73] M. Clerc and J. Kennedy, "The particle swarm — explosion, stability, and convergence in a multidimensional complex space", *IEEE Trans. on Evolutionary Computation*, vol. 6, no. 1, pp. 58-73, 2002.

[74] S. U. Guan, Q. Chen and W. T. Mo, "Evolving for Dynamic Multi-objective Optimization Problems with Objective Replacement", *Artificial Intelligence Review*, vol. 23, pp. 267-293, 2005.

[75] J. Kennedy, R. C. Eberhart and Y. H. Shi, *Swarm Intelligence*. Morgan Kaufmann Publishers, San Francisco, 2001.

[76] T. M. Blackwell and P. J. Bentley, "Dynamic Search with Charged Swarms", In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 19-26, 2002.

[77] M. Løvbjerg, T. K. Rasmussen and T. Krink, "Hybrid Particle swarm optimiser with breeding and subpopulations", In *Proceedings of the genetic and*

evolutionary computation conference (GECCO), San Francisco, USA, July, 2001.

[78] F. Wu, *A Framework for Memetic Algorithms*. MSc. Thesis, Department of Computer Science, University of Auckland, New Zealand, 2001.

[79] A. E. Conradie, R. Miikkulainen and C. Aldrich, "Intelligent Process Control utilizing Symbiotic Memetic Neruo-Evolution", In *Proceedings of the 2002 Congress on Evolutionary Computation*, vol. 1, pp. 623-628, 2002.

[80] T. Bäck, F. Hoffmeister and H. P. Schwefel, "A Survey of Evolution Strategies", In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pp. 2. Morgan Kaufmann, San Mateo, CA. 1991.

[81] H. P. Schwefel, *Evolution and optimum seeking*, John Wiley & Sons, New York. 1995.

[82] D. H. Wolpert and W. G. Macready, "No Free Lunch Theorems for Optimization", *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 4, pp. 67-82, 1997.

[83] S. Christensen and F. Oppacher, "What can we learn from No Free Lunch? A First Attemp to Characterize the Concept of a Searchable Function.", In *Proccedings of the Genetic and Evolutionary Computation Conference*, pp. 1219-1226. San Franscisco, USA, July, 2001.

[84] Y. H. Shi and R. C. Eberhart, "Empirical Study of Particle Swarm Optimization", In *Proceedings of the Congress of Evolutionary Computation*, vol. 3, pp. 1945C1950, IEEE Press, 1999.

[85] Q. Chen and S.U. Guan, "Incremental Multiple Objective Genetic Algorithms." *IEEE Trans. Syst., Man, Cybern. B*, vol. 34, no. 3, pp. 1325-1334, Jun. 2004.

[86] E. Zitzler and L. Thiele, "Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach", *IEEE Trans. Evolut. Comput.*, vol. 3, no. 4, pp. 257-271, Nov. 1999.

[87] E. Zitzler and L. Thiele, "An evolutionary algorithm for multi-objective optimization: the strength rateto approach", *TIK- Report*, Swiss Federal Institute of Technology, no. 43, 1998.

[88] E. Zitzler, *Evolutionary Algorithms for Multiobjective Optimization. Methods and Applications*. Swiss Federal Institute of Technology (ETH), Zurich. Shaker Verlag, Germany, Dec. 1999.

[89] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results", *Evolutionary Computation*, vol. 8, no. 2, pp. 173-195, Apr. 2000.

[90] J. D. Knowles and D. W. Corne, "On metrics for comparing nondominated sets. In congress on evolutionary computation," In *Proc. Congress on Evolut. Comput. (CEC02)*, vol. 1, pp. 711C716, Piscataway, NJ, 2002.

[91] J. D. Knowles, D. W. Corne, and M. J. Oates et al., "On the assessment of multiobjective approaches to the adaptive distributed database management problem," in *Parallel Problem Solving From NaturePPSN VI*, M. Schoenauer et al., Eds. Berlin, Germany: Springer-Verlag, pp. 869C878, 2000.

[92] M. P. Hansen and A. Jaszkiewicz, "Evaluating the Quality of Approximations to the Nondominated Set," *Tech. Rep.*, Institute of Mathematical Modeling Technical University of Denmark, IMM-REP-1998-7, 1998.

[93] A. Farhang-Mehr and S. Azarm, "Diversity assessment of Pareto optimal solution sets: An entropy approach," In *Proc. Congress Evolut. Comput.*, vol. 1, pp. 723C728, 2002.

[94] E. Zitzler and L. Thiele, "Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach," *IEEE Trans. on Evolut. Comput*, vol. 3, no. 4, pp. 257-271, Nov. 1999.

[95] E. Zitzler and L. Thiele, "An evolutionary algorithm for multi-objective optimization: the strength Pareto approach," *TIK-Report*, Swiss Federal Institute of Technology, no. 43, 1998.

[96] J. D. Knowles and D. W. Corne, "Approximating the nondominated front using the pareto archived evolution strategy," *Evolutionary Computation*, vol. 8, no. 2, pp. 149-172, 2000.

[97] M. Laumanns, G. Rudolph and H. P. Schwefel. "A spatial predator-prey approach to multi-objective optimization," *Parallel Problem Solving from Nature*, vol. 5, pp. 241-249, 1998

[98] C. M. Fonseca and P. J. Fleming, "Multiobjective optimization and multiple constraint handling with evolutionary algorithms. I. A unified formulation," *IEEE Trans. on Syst., Man and Cybern. A*, vol. 28, no. 1, pp. 26 -37, Jan. 1998.

[99] C. M. Fonseca and P. J. Fleming, "Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization, In *Proc. of the Fifth Intl. Conf. on Gen. Algo.*, pp. 416-423, San Mateo, California, July 17-21 1993.

[100] J. D. Schaffer, "Multiple Objective Optimization with Vector Evaluated Genetic Algorithms," *Genetic Algorithms and their applications: Proceedings of the First International Conference on Gen. Algo.*, pp. 93-100, Lawrence Erlbaum, 1985.

[101] F. Kursawe, "A variant of evolution strategies for vector optimization," *Parallel Problem Solving from Nature*, pp. 193-197, Springer, 1991.

[102] S. U. Guan and S. C. Li, "Incremental Learning with Respect to New Incoming Input Attributes" *Neural Processing Letters*, pp. 241-260, vol. 14, no. 3, Dec. 2001.

[103] S. U. Guan and J. Liu, "Incremental Ordered Neural Network Training," *Journal of Intelligent Systems*, vol. 12, no. 3, pp. 137-172, 2002.

[104] S. U. Guan and P. Li, "A Hierarchical Incremental Learning Approach to Task Decomposition," *Journal of Intelligent Systems*, vol. 12, no. 3, pp. 201-226, 2002.

[105] S. U. Guan and F. M. Zhu, "Incremental Learning of Collaborative Classifier Agents with New Class Acquisition — An Incremental Genetic Algorithm Approach," *International Journal of Intelligent Systems*, vol. 18, no. 11, pp. 1173-1193, Nov. 2003.

[106] S. U. Guan and J. Liu, "Incremental Neural Network Training with an Increasing Input Dimension," *Journal of Intelligent Systems*, vol. 13, no. 1, pp. 45-70, 2004.

[107] S. U. Guan and P. Li, "Incremental Learning in Terms of Output Attributes," *Journal of Intelligent Systems*, vol. 13, no. 2, pp. 95-122, 2004.

[108] C. M. Fonseca and P. J. Fleming, "An Overview of Evolutionary Algorithms in Multiobjective Optimization," *Evolutionary Computation*, 3(1), pp. 1-16, Spring 1995.

[109] H. A. Taha, *Operations Research: An Introduction*. Prentice-Hall, Singapore, 2003.

[110] Q. Chen, *Objective Increment, Its Effect and Application in Multi-Objective Optimization Evolution*, Master Thesis, National University of Singapore, 2003.

[111] R. C. Purshouse and P. J. Fleming, "Conflict, Harmony, and Independence: Relationships in Evolutionary Multi-Criterion Optimisation", In *Proc. of Evolutionary Multi-criterion Optimization*, pp. 8-11, Apr., 2003.

[112] P. Schroder, *Multivariable Control of active magnetic bearings*, Doctor dissertation, University of Sheffield, 1998.

[113] K. Deb, "Single and Multi-Objective Optimization Using Evolutionary Computation." *Technical Report, 2004002*, KanGAL, IIT, India 2004.

[114] S. Kukkonen and J. Lampinen, "An Empirical Study of Control Parameters for The Third Version of Generalized Differential Evolution (GDE3)", *IEEE Congress on Evolutionary Computation (CEC)*, pp. 2002-2009, July, 2006.

[115] J. Knowles, "ParEGO: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems", *IEEE Transactions on Evolutionary Computation*,Vol. 10, No. 1, pp. 50-66, 2006.

[116] A. Gaspar-Cunha and J. A. Covas, "A Real-World Test Problem for EMO Algorithms", In *Proceedings of Evolutionary Multi-Criterion Optimization: Second International Conference*, Faro, Portugal, April, 2003.

[117] J. Knowles and D. Corne, "On metrics for comparing nondominated sets", In Proceedings of the 2002 Congress on Evolutionary Computation (CEC'02), vol. 1, pp. 711-716, May, 2002.

[118] Q. Chen and S.U. Guan, "Incremental Multiple Objective Genetic Algorithms", *IEEE Transactions on System, Man, Cybernetics B*, vol. 34, no. 3, pp. 1325-1334, Jun. 2004.

Appendix A

Preprocessing Procedure for Tuning Parameters

Normally, the parameters for heuristic optimization algorithms are set according to experiences [114]. In my study, a preprocessing procedure is used to tune the parameters at the beginning of each experiment. Configuration guidelines are obtained from the results of this procedure. The pseudo-code of the preprocessing procedure is described as follows:

1. Fix the values of all the parameters according to those suggested in related work.
2. Change one of the parameters with others being fixed and run simulations.
3. Adjust the dimensionality of the problem if necessary and redo step 2.

With the results obtained from the procedure described above, we are able to observe how the performance varies with the change of the parameters and dimensionality. The findings may help us choose a set of parameters that results

in a satisfactory outcome with limited computational time. It is important to be noted that the obtained parameter set is a tradeoff between good convergence and cost.

Appendix B

Computational Complexity of Algorithms Used in the Study

In this study, all the experiments are conducted on a Pentium IV 2.0GHz PC with 1G RAM. The performance of different algorithms is compared on the time they take to solve a certain problem. Their computational complexity is estimated and shown in the following tables.

In Table B.1, g is the number of generations, m is the population size of the algorithms used to solve SOPs in the study.

Table B.1: Computational Complexity of Algorithms Solving SOPs

Algorithms	IPSO	ASPSO	CSPSO	HPSO_BS
Complexity	$O(gm)$	$O(gm)$	$O(gm^2)$	$O(gm)$

As shown in this table, the computational complexities of IPSO, ASPSO and HPSO_BS are the same, while that of CSPSO is higher in terms of the population size. This is because the repulsion between any two particles needs to be computed in every generation.

In Table B.2, g is the number of generation, n is the number objectives, M is the population size of the algorithms used to solve MOPs in the study.

Table B.2: Computational Complexity of Algorithms Solving MOPs

Algorithms	IMOPSO	MOPSO	IMOGA	SPEA	NSGA-II
Complexity	$O(gnM)$	$O(gnM)$	$O(gnM^2)$	$O(gnM^2)$	$O(gnM^2)$

As shown in the table above, the computational complexities of IMOPSO and MOPSO are the same, which is lower than that of IMOGA, SPEA and NSGA-II. The reason why those GA-based algorithms has a higher computational complexity is that the objective values of the individuls in a population must be ranked, which is not required in PSO-based algorithms.